**UNIVERSIDAD CARLOS III DE MADRID**
**ESCUELA POLITÉCNICA SUPERIOR**


**INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN**
**PROYECTO FIN DE CARRERA**

# IMPLEMENTACIÓN DE PROTOCOLO DE INTERCAMBIO JUSTO BASADO EN POLÍTICAS DE FIRMA EXTENDIDAS: OFEPSP+

**Alumno:** Fernández Nuevo, Miguel Ángel
**Tutor:** González-Tablas Ferreres , Ana Isabel
**Co-director:** López Hernández-Ardieta, Jorge
**Año:** 2015

# Contenido

# Lista de figuras

# Lista de tablas

**TRIBUNAL**

Presidente:

Secretario:

Vocal:

Realizado el acto de defensa del Proyecto Fin de Carrera el día de de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Fdo. Presidente Fdo. Secretario Fdo. Vocal

# Capítulo 1. Introducción

## 1.1 Descripción

En el presente proyecto se pretende realizar la implementación necesaria para ejecutar un protocolo de intercambio justo basado en políticas de firma extendidas, basándose en el protocolo OFEPSP+ [JLHA]. La idea principal es realizar una implementación de los actores implicados: Origen, Receptor del intercambio, y tercera entidad de confianza.

Se implementa un prestador de servicios de certificación que publica los servicios esenciales necesarios para poder ejecutar el protocolo: emisión de certificados, métodos de validación por CRL y OCSP, y generación y publicación de Políticas de Firma y Políticas de Firma Extendidas.

La implementación del protocolo y su ejecución permitirán comprobar la viabilidad de su implantación.

## 1.2 Acrónimos

| | |
|---|---|
| ASN.1 | *Abstract Syntax Notation One,* notación sintáctica abstracta uno |
| B2B | *Business-to-Business* |
| B2C | *Business-to-Consumer* |
| CA | *Certification Authority,* autoridad de certificación |
| CRL | *Certificate Revocation List,* lista de revocación de certificados |
| EPES | *Explicit Policy based Electronic Signature*, firma electrónica basada en políticas indicadas de modo explícito |
| ETSI | *European Telecommunications Standards Institute*, instituto europeo de estándares en las telecomunicaciones |
| extSP | *Extended Signature Policy,* Política de Firma Extendida |
| HEX | Hexadecimal |
| JAXB | *Java Architecture for XML Binding* |
| JSF | *Java Server Faces* |
| NRE | *Non-repudation evidence* |
| OCSP | *Online Certificate Status Protocol* |
| OFEPSP+ | *Optimistic Fair Exchange Protocol based on Signature Policies,* protocolo de intercambio justo optimista basado en políticas de firma |
| ORM | *Object-Relational mapping*, mapeo Objeto-Relacional |
| PKCS | *Public-Key Cryptography Standards* |
| PKI | *Public Key Infrastructure*, infraestructura de clave pública |
| PNRO | *Partial non-repudation evidence of origin* |
| PNRR | *Partial non-repudation evidence of receipt* |
| POJO | *Plain Old Java Object,* objeto java plano tradicional |
| POM | *Project Object Model* |
| PSC | Prestador de Servicios de Certificación |
| SP | *Signature Policy,* Política de Firma |
| TR | *Technical Report*, informe técnico |
| WAR | *Web Archive*, archivo Web |
| XAdES | *XML Advanced Electronic Signatures,* firma electrónica avanzada XML |
| XML | *Extensible Markup Language,* lenguaje de marcas extensible |
| XSD | *XML Schema Definition* |

*Tabla 1: Acrónimos*

## 1.3 Estructura del documento

A continuación se indica los puntos desarrollados en los capítulos de la presente memoria:

Capítulo 1: Se pretende dar una visión global del proyecto realizado, listado de acrónimos utilizados y la estructura de la presente memoria.

Capítulo 2: En el Estado de la Cuestión se tratan temas de firma electrónica hasta la descripción de políticas de firma extendidas, así como los escenarios de intercambio justo y el propio protocolo a implementar.

Capítulo 3: Se describe el análisis funcional y diseño técnico a alto nivel de cada uno de los módulos implementados para ejecutar el protocolo. Del mismo modo se describe el modo de generar y ejecutar dicho protocolo.

Capítulo 4: Se describen las conclusiones obtenidas de la implementación así como las mejoras y trabajos futuros.

Capítulo 5: Anexos incluidos para describir y aclarar los elementos descritos en la implementación del protocolo.

# Capítulo 2. Estado de la Cuestión

Uno de los principales requisitos del protocolo a implementar será garantizar la integridad, autenticidad y el no repudio de los datos intercambiados entre las distintas entidades, requisitos que se cumplen con la aplicación de firmas electrónicas. Para una mejor comprensión de este documento, en esta introducción se definirán los conceptos básicos de cifrado y firma electrónica que permitirán entender las bases y funcionamiento del protocolo implementado.

## 2.1 Infraestructura de Clave Pública y Firma Electrónica

### 2.1.1 Sistemas de cifrado

El cifrado es un procedimiento que, mediante el uso de un determinado algoritmo de cifrado con cierta clave de cifrado, transforma un conjunto de datos de tal forma que sea incomprensible a toda persona que no tenga la clave secreta (clave de descifrado) del algoritmo que se usa para poder descifrarlo (algoritmo de descifrado).

De esta forma podemos señalar la existencia de dos algoritmos: algoritmo de cifrado y algoritmo de descifrado, y la existencia de dos claves: clave de cifrado y clave de descifrado. En los casos en que la clave de cifrado es la misma que la de descifrado, hablamos de algoritmos de cifrado simétricos; por otro lado, si las claves son distintas se denominan algoritmos de cifrado asimétricos.

#### *2.1.1.1Algoritmos de cifrado simétrico*

En los algoritmos de cifrado simétrico se utiliza la misma clave para cifrar y para descifrar. A esta clave la denominamos Clave Maestra. Tanto el remitente del mensaje como el destinatario del mismo deberán conocer dicha Clave Maestra.

La problemática de los algoritmos de cifrado simétrico es la necesidad de comunicar la Clave Maestra. Para solucionar este problema surgieron los algoritmos de cifrado asimétrico.

Algunos algoritmos de técnicas de clave simétrica son: DES, 3DES, Blowfish, AES (Rijndael).

#### *2.1.1.2Algoritmos de cifrado asimétrico*

Un cifrado de clave pública (o asimétrica) se basa en el uso de una pareja de claves: Clave Pública y Clave Privada, de las cuales una se usa para cifrar y la otra para descifrar.

Todo lo que se requiere es que, antes de iniciar la comunicación secreta, el remitente consiga una copia de la clave pública del destinatario. Es más, esa misma clave pública puede ser usada por cualquiera que desee comunicarse con su propietario.

Cuando un usuario desea enviar un mensaje a otro usuario, sólo debe cifrar el mensaje que desea enviar utilizando la clave pública del receptor. Por su parte, el receptor podrá descifrar el mensaje con su clave privada que solo él conoce. Algunos algoritmos de técnicas de clave asimétrica son: Diffie-Hellman, RSA, DSA, ElGamal, Criptografía de curva elíptica.

### 2.1.2 Infraestructura de clave pública

PKI (*Public Key Infrastructure*, en castellano, Infraestructura de Clave Pública) se refiere a un grupo de soluciones técnicas que permiten la aplicación de la criptografía de clave pública en entornos inseguros como es Internet. Sus funciones son múltiples y abarca, entre otras, las áreas siguientes:

- Generar pares de claves (clave pública/clave privada) garantizando la confidencialidad de las claves privadas.
- Registrar las solicitudes de claves a través de la verificación de la identidad de los solicitantes.
- Certificar la relación entre el usuario y su clave pública.
- Revocar claves, en caso de que su propietario la haya perdido, en caso de que el período de validez haya expirado o se encuentre en riesgo.

### 2.1.3 Certificado electrónico

Un certificado electrónico es un componente basado en sistemas de clave pública que identifica de manera telemática a una persona física o jurídica, a una organización o a un sistema informático. Los certificados electrónicos siguen el estándar X.509 [RFC3280] y son emitidos por una Infraestructura de clave pública (PKI) que garantiza la equivalencia “certificado <-> persona/Entidad/Sistema”.

Podemos decir que un Certificado Digital es el equivalente electrónico a un Documento de Identidad, asociando una clave criptográfica a una identidad, de tal forma que ésta quede fehacientemente ligada a los documentos electrónicos sobre la que se aplica. Un Certificado Electrónico sirve básicamente para:

- Autentificar la identidad del usuario de forma electrónica ante terceros.
- Firmar digitalmente de forma que se garantice la integridad de los datos trasmitidos y su procedencia.
- Cifrar datos para que solo el destinatario del documento pueda acceder a su contenido.

Su uso garantiza:

- La identidad del emisor y del receptor de la información (autentificación de las partes).
- Que el mensaje no ha sido manipulado durante el envío (integridad de la transacción).
- Que solo emisor y receptor vean la información (confidencialidad).
- Que el titular de un mensaje no pueda negar que efectivamente lo firmó (no-repudio).

Los certificados electrónicos según [RFC3280] básicamente se componen de:

- Versión: identifica el formato del certificado.
- Nº de serie: cada certificado es único dentro de la Autoridad Certificadora.
- Algoritmo: se pueden utilizar múltiples algoritmos para firmar el certificado.
- Autoridad Emisora: nombre de la Autoridad de Certificación que emite el certificado.
- Periodo de validez: desde / hasta.
- Asunto: Nombre del usuario y NIF o NIE.
- Firma de la Autoridad de Certificación.

Además, los certificados electrónicos están constituidos por un par de claves (pública y privada) que se suelen almacenar en almacenes de certificados (Keystores). Los almacenes más comunes suelen ser:

- Tokens criptográficos Hardware, como tarjetas criptográficas (SmartCards) o dispositivos USB.
- Almacenes de Certificados de los navegadores, como Internet Explorer o Mozilla.
- Los certificados electrónicos pueden estar almacenados en ficheros para su transporte o para hacer copias de seguridad de los mismos, como por ejemplo ficheros en formato PKCS#12.

### 2.1.4 Firma electrónica

Una firma electrónica es el resultado de aplicar una serie de operaciones criptográficas a una fuente de información (por ejemplo, un documento) utilizando un certificado electrónico. De esa forma consigue garantizar:

- **Integridad del documento firmado**: Garantía de que una información, o un conjunto de datos en general, no es modificado en la transmisión desde el generador al receptor.
- **No repudio**: Garantía de que el emisor es el autor de la transacción que ha sido firmada.

El procedimiento de firmado se resume a continuación:

1. Se calcula un resumen, conocido como código hash, a partir del documento original. Este código hash puede calcularse mediante varios algoritmos (SHA1, SHA2, MD2, MD5, etc.) y garantiza de manera unívoca la relación código hash <-> documento. De esta forma, una ligera alteración en un documento (por ejemplo, una coma) cambiaría el código hash calculado previamente.
2. Haciendo uso de la clave privada asociada a un certificado electrónico se cifra el código hash calculado anteriormente, mediante un algoritmo asimétrico (por ejemplo, RSA). Este proceso

se realiza en el entorno del usuario que firma, para así acceder a la clave privada del mismo (en navegador o tarjeta criptográfica). En la firma electrónica se cifra con la clave privada (suele requerir PIN) y se descifra (proceso de verificación) con la pública del firmante.

3. Se genera una estructura de firma electrónica que contiene entre otras cosas: certificado con la clave pública, código hash cifrado, información del firmante, algoritmos utilizados, etc.

#### *2.1.4.1Tipología de Firma Electrónica*

No existe una única clasificación de la firma electrónica, sino que existen varias clasificaciones posibles en función del criterio utilizado. A continuación se muestran varias clasificaciones posibles.

Según el Formato (entendiendo el formato como la estructura de firma electrónica utilizada). Existen varios formatos estándares que son mantenidos por organismos internacionales y de reconocido prestigio. Ejemplos de éstos son:

- La serie de formatos en XML, **XAdES** (*XML Advance Electronic Signature*)
- Formatos evolucionados a partir de CMS, **CAdES** (*CMS Advance Electronic Signature*)

Por otro lado, existen formatos de firma electrónica basados en formatos de documentos muy extendidos en el mercado. Ejemplos de estos formatos son:

- **Firma en PDF:** La firma en este formato es propietaria de Adobe y por desgracia no es compatible hacia atrás en todas sus versiones. No obstante, incorpora la firma en el propio documento, la representación gráfica es bastante buena y está bastante extendido.
- **Firma en ODF** (*Open Document Format*): El formato ODF es utilizado por los paquetes ofimáticos OpenOffice y StarOffice. Sigue la misma pauta de comportamiento que el formato PDF.
- **Firma en OOXML** (*Office Open XML*): Especificación definida originalmente por Microsoft para reemplazar sus formatos binarios protegidos, en la segunda parte de la normativa ISO/IEC 29500 se definen las propiedades principales de la Firma Electrónica en OOXML.

Según la ubicación de la información firmada:

- **Explícita**. Cuando la estructura de firma electrónica es independiente del documento firmado, es decir, en el proceso de generación de firma se obtienen dos ficheros: uno con la firma y otro con el documento original. Este tipo es el más utilizado, sobre todo en documentos de tamaño mediano-grande.
- **Implícita**. Cuando la estructura de firma electrónica incorpora el documento firmado. Este formato se utiliza cuando los documentos a firmar son pequeños (por ejemplo las peticiones a servicios Web).

Según la multiplicidad de la firma electrónica:

- **Simple**. Cuando en la estructura de firma electrónica solo existe un único firmante.
- **Multifirma**. Cuando en la estructura de firma electrónica existen varios firmantes.

Según la jerarquía de firma electrónica:

- **Jerárquica** o *counterSignature*. Cuando la estructura de firma electrónica es secuencial y en cadena, es decir, antes de firmar una determinada persona ha de firmar otra previamente. Lo

que realmente se firman son las firmas anteriores manifestando de alguna forma la conformidad con lo firmado.

- **Paralela** o *coSign*. Cuando en la estructura de firma electrónica no existe ni orden ni jerarquía, lo que realmente importa es que un conjunto de "n" personas firmen un mismo documento.

Como se puede ver, la clasificación de las firmas electrónicas es muy diversa y además se puede cruzar entre los diferentes criterios de clasificación. De esta forma, una firma puede ser en formato CMS, explícita, *counterSignature y* multifirma. Los criterios comentados son los más extendidos, aunque existen varios más que se han omitido para simplificar esta clasificación.

### *2.1.4.2Validación de firmas electrónicas y sellado de tiempo*

El proceso de validación de una firma electrónica podemos dividirlo en tres fases:

1. **Validación básica**. Se verifica la integridad de la firma electrónica y que los datos firmados se corresponden con el original aportado en el proceso de validación. Este proceso de validación se resume a continuación:
    a. Se utiliza la clave pública contenida en la estructura de firma para descifrar el hash cifrado en la firma.
    b. Se calcula el código hash del documento, utilizando el mismo algoritmo que en el proceso de firma anterior.
    c. Se comprueba que el código hash obtenido en el paso a) y en el b) son idénticos. En el caso de que no lo sean, significa que el documento o la firma electrónica han sido manipulados con lo que la firma electrónica es inválida.

2. **Validación del certificado** empleado en la firma electrónica. Este proceso conlleva la validación del certificado electrónico contra el prestador de servicios de certificación correspondiente. Para realizar la validación del estado de revocación de un certificado es necesario consultar la lista de certificados revocados (CRL) publicada por el prestador o acceder a un servicio OCSP, este último proporciona una información más adecuada y reciente del estado de revocación de los certificados.

3. **Validación del sello de tiempo**. Como es sabido, todo certificado tiene asociado un estado de validez. Si una firma electrónica es realizada con un certificado válido en un instante de tiempo "X" y en el instante "X+Y" el certificado utilizado está revocado, al verificar la firma se verificaría también el certificado empleado obteniendo como resultado que está revocado y que la firma es incorrecta. Para evitar este inconveniente, todo proceso de firma electrónica necesita un elemento externo que permita verificar la validez de la firma: la fecha exacta en que se realizó. Cuando se verifica una firma y se obtiene que el certificado está revocado, se comprueba si la fecha de revocación del certificado empleado es posterior a la fecha en que se realizó la firma, en cuyo caso se constata que el certificado no estaba revocado en ese momento y la firma es válida. Por este motivo, se recomienda que toda firma electrónica tenga un fechado electrónico, denominado técnicamente como *TimeStamping*, obtenido de una fuente de tiempos segura que certifique que ese instante de tiempo es seguro e inalterable. La entidad que certifica que el instante de tiempos es seguro se denomina Autoridad de Fechado Electrónico (*TimeStamping Authority, TSA*) [RFC3161].

## 2.2 Políticas de Firma

Los elementos principales que hemos visto hasta este momento (la existencia de múltiples certificados, la proliferación de formatos, procedimientos y métodos de firma y de validación de la misma y la problemática del significado y garantías de la firma) hacen notar la necesidad de gestionar las firmas electrónicas. Para ello, y bajo los auspicios del grupo Iniciativa Europea de Normalización de la Firma Electrónica, impulsado por la Comisión Europea, el Instituto Europeo de Normas de Telecomunicaciones (ETSI) ha producido una especificación técnica, [TR102041], que trata de forma monográfica la gestión de las firmas electrónicas mediante políticas de seguridad especialmente diseñadas al respecto, denominadas políticas de firma electrónica.

Una política de firma electrónica es un documento legal que contiene las normas relativas para la creación y validación de una firma electrónica, de acuerdo con las cuales se puede considerar que una firma electrónica es válida. Se trata de lograr que el firmante y el verificador empleen la misma normativa y que esta política de firma regule las condiciones en las que las partes confían en una firma electrónica para un contexto de seguridad dado.

Cada política de firma se compone de un conjunto de reglas que explicitan, de cara al firmante, qué información debe incluir en la Firma Electrónica a la hora de generarla y, de cara al verificador de la firma, qué información debe tener en cuenta para llevar a cabo el proceso de validación de la Firma Electrónica.

Además, todos estos procesos se engloban en un contexto contractual/jurídico/legal dentro del cual la Firma Electrónica tiene o no validez. De esta forma podemos decir que una política de firma electrónica es un conjunto de reglas que se emplean en la creación y en la validación de una firma electrónica, de acuerdo con las cuales se puede considerar que una firma electrónica es válida completando las carencias legales, mediante una autorregulación técnico-jurídica.

Existen por tanto tres actores en un escenario de Firma Electrónica con Política de Firma: el firmante, el verificador y el emisor de la Política de Firma. Éste último tiene la competencia de generar y gestionar el documento de política de firma al cual se adhieren tanto el firmante como el verificador.

Una política de firma puede estar referenciada en una Firma Electrónica de forma explícita (mediante un atributo firmado en la propia estructura de la firma) o implícita, a través de la semántica del mensaje firmado. Es conveniente que la política de firma se encuentre disponible en formato legible de forma que se aseveren los requerimientos legales y contextuales en los cuales se aplica. Por otro lado, se necesita incluir información de la política de firma procesable automáticamente (en formato ASN.1 [TR102271] o XML [TR102038]), para facilitar los mecanismos de creación y verificación de las firmas adheridas a dicha política.

A grandes rasgos, una política de firma engloba los siguientes aspectos [TR102041]:

- Información de la Política de Firma (Información general):
  - Identificador único de la política de firma.
  - Nombre del emisor y fecha de expedición de la política de firma.
  - Campo de aplicación de la política de firma.

- Política de validación de firma:
  - Periodo de validez de aplicación de la política de firma.
  - Lista de tipos de compromisos reconocidos.
  - Reglas para el Uso de Autoridades de Certificación.
  - Reglas para el Uso de Información de Estado de Revocación.
  - Reglas para el Uso de Roles.
  - Reglas para el Uso de Sellados de Tiempo y Marcas de Tiempo.
  - Datos de verificación de firma a proporcionar por el firmante.
  - Datos de verificación de firma a recuperar por parte del verificador.
  - Restricciones en los algoritmos de firma y longitudes de clave empleados.

- Extensiones que se necesiten incorporar en la Política de Firma.

Como ejemplos de Políticas de Firmas existentes, en el ámbito de la Administración General del Estado se encuentran publicadas las siguientes:

| **Emisor/Ámbito** | **Administración General del Estado** | **Ministerio de Industria, Energía y Turismo / Facturación Electrónica** |
|---|---|---|
| **Formato Legible** | Publicada en [POLAGE] | Publicada en [POLFE] |
| **Implementación XML** | Publicada en [POLAGE] | Publicada en [POLAGE] |
| **Implementación ASN1** | Publicada en [POLAGE] | No aplica al permitir unicamente firmas XML |

*Tabla 2: Ejemplos de Políticas de Firma publicadas*

En la Resolución de 19 de julio de 2011, de la Secretaría de Estado para la Función Pública, se aprueba la Norma Técnica de Interoperabilidad de Política de Firma Electrónica y de certificados de la Administración. En esta resolución se define el contenido de la Política de Firma aplicable en el ámbito de la Administración General del Estado [POLAGE], para posteriormente publicarse dicha política en formato legible, así como en XML y ASN.1. En esta Política de Firma, en sus reglas comunes generales por ejemplo, se define como requisito que los formatos de firma permitidos son XAdES, CAdES y PAdES de determinadas versiones. Al aplicarse esta Política de Firma, por ejemplo

si se genera una firma XMLDSignature, esta no es válida dentro del ámbito de la Administración General del Estado ya que no cumple con las reglas de la Política de Firma.

Otro ejemplo de Política de Firma actualmente publicada y en uso es la Política de Firma de Facturae [POLFE], en la que se definen las reglas de generación y validación de las firmas de facturas electrónicas en formato Facturae. Como ejemplo, esta Política de Firma define como requisito que las firmas generadas de una factura electrónica deben realizarse con formato XAdES de una determinada versión. De este modo, si un proveedor emite una factura en formato Facturae a un organismo de la Administración Pública firmada con un formato distinto a XAdES (por ejemplo CAdES o PAdES), dicha factura no es admitida por el receptor ya que su firma no cumple con la Política de Firma.

De la existencia de estas dos Políticas de Firmas surge la problemática de tener una Política de Firma para un ámbito muy amplio, como es la Administración General del Estado (AGE), y a su vez una Política de Firma más especifica (Facturae) que se encuentra en el mismo ámbito. Para solucionar esto, dentro de la propia Política de Firma de la AGE se define un **nivel de compromiso** con las reglas aplicables solo en los casos de facturación electrónica.

## 2.3 Políticas de Firma Extendidas

Como se ha descrito en el apartado anterior, las Políticas de Firma definen las normas a seguir para generar firmas y las condiciones para que sea válida en un ámbito determinado, pero no permite la gestión de un conjunto de firmas generadas en una transacción. Para cubrir esta necesidad se definen las Políticas de Firma Extendidas (extSP) descritas en el capítulo 8 de [JLHA].

En resumen, una política de firma extendida permite la gestión de un conjunto de firmas generadas en una transacción, y se compone de:

- Información general de la Política de Firma Extendida:
  - Identificador único de la política de firma extendida.
  - Nombre del emisor y fecha de expedición.
  - Otras propiedades opcionales (por ejemplo lugar de publicación).

- Política de validación:
  - Periodo de validez en el que puede aplicarse la política.
  - Árboles de soluciones. Descrito en apartado 2.3.1.
  - Descripción del contexto en el que se aplica la política de firma extendida. Descrito en apartado 2.3.2.

- Extensiones que se necesiten incorporar en la política de firma extendida.

### 2.3.1 Árboles de Solución

Dentro de las políticas de validación de las extSP se definen un conjunto de gráficos donde cada uno representa un árbol de firmas en el que se incluyen las dependencias y relaciones entre estas. De esta forma, cada árbol existente en la extSP representa un conjunto de firmas de una transacción, donde las firmas en el nivel superior del árbol se corresponde con firmas primarias, que son o bien firmas en paralelo o firmas secuenciales; y el resto de las firmas corresponden a *counterSignatures*, que se pueden aplicar ya sea a una firma primaria simple u a otra *counterSignatures*.

Las firmas existente en los arboles de solución, además de indicar la relación entre ellas, incluyen información adicional a utilizar en los procesos de validación. Esta información básicamente se compone de los tiempos de creación aceptables entre firmas, y los roles, SP y niveles de compromiso que aplican en cada una. La definición y requisitos de los árboles de solución se encuentran descritos en el apartado 8.1 de [JLHA].

### 2.3.2 Contexto de Aplicación

Además de los Árboles de Solución, dentro de las políticas de validación se determina el contexto en el cual es aplicable la extSP. En este elemento se define el ámbito en el que la aplicación de la política de firma extendida es adecuado, por ejemplo Administración Electrónica, Comercio Electrónico, Servicios Electrónicos de Salud, etc. Y también se incluye información adicional sobre el contexto de las transacciones, por ejemplo intercambios de documentación con información sensible, firma de contratos, operaciones de compra on-line, etc.

La información indicada en el contexto de aplicación de la política de validación, junto al contexto indicado en cada SP, permiten determinar si las políticas utilizadas por las partes involucradas son las adecuadas dentro de una transacción determinada, sobre todo si es necesaria la resolución de disputas. La forma de definir el contexto de aplicación en las extSP se encuentra descrito en el apartado 8.1 de [JLHA].

### 2.3.3 Ejemplo de aplicación

A diferencia de las Políticas de Firma, no existen publicadas Políticas de Firma Extendidas. De hecho uno de los objetivos del presente proyecto es implementar un escenario en el que se utilicen (protocolo de intercambio justo OFEPSP+) y demostrar de este modo su utilidad y la viabilidad de su implantación.

Las Políticas de Firma Extendidas son necesarias en transacciones en las que se realizan diversas firmas electrónicas y se determinan a priori las firmas necesarias y la relación entre ellas dentro de cada transacción. Por ejemplo, en un escenario en el que un arrendatario y un arrendador realizan la

firma de un contrato de arrendamiento, y cada uno tiene que dejar constancia de aceptar las condiciones del contrato. Se podría definir una Política de Firma Extendida para un Contexto de Aplicación de firma de contratos de arrendamiento, con un Árbol de Solución con las firmas y su relación. Según este ejemplo de aplicación de Políticas de Firma Extendidas este árbol podría ser el siguiente:

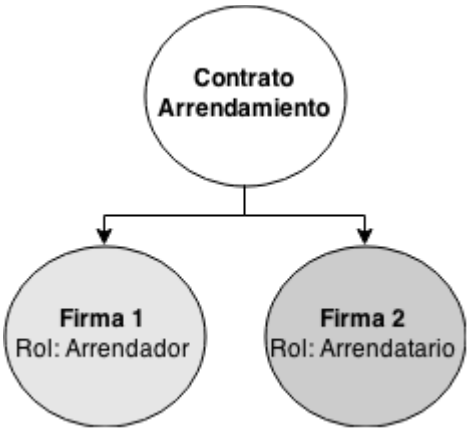



*Figura 1: Ejemplo Árbol de Solución*

El nodo raíz del árbol representa el Contrato de Arrendamiento. El nodo "Firma 1" indica que el arrendador debe firmar el Contrato de Arrendamiento para dejar constancia de su conformidad con las condiciones del contrato. Del mismo modo, el nodo "Firma 2" se corresponde con la firma del contrato a realizar por el arrendatario.

Continuando con este ejemplo, la Política de Firma Extendida podría obligar a que en la transacción de arrendamiento, tanto el arrendador como el arrendatario estén obligados a dejar constancia de que han sido informados de que el otro ha aceptado las condiciones del contrato. Para ello, en el Árbol de Solución se podrían incluir dos nuevos nodos.

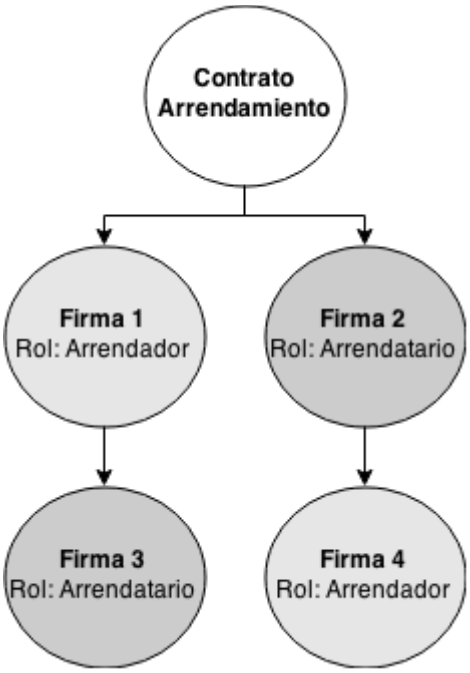



*Figura 2: Ejemplo Árbol de Solución con información de aceptación*

En este ejemplo, las firmas 3 y 4 corresponden a *counterSignatures* realizadas por el arrendatario y arrendador respectivamente, dejando constancia de que ambos han sido informados de que el otro ha aceptado las condiciones del contrato.

Dentro de este Árbol de Solución de ejemplo también se podrían establecer otras condiciones en la relación entre los nodos de firma; por ejemplo, se podría definir un tiempo de caducidad de la oferta de arrendamiento, incluyendo una condición de tiempo máximo de generación de la Firma 2 respecto la Firma 1.

## 2.4 Protocolos de Intercambio Justo

Los Protocolos de Intercambio Justo son un mecanismo que pretende asegurar un intercambio de información sin que ninguna de las entidades involucradas posea en ningún momento una ventaja sobre la otra. La implantación de estos protocolos sería trivial si se parte de que todas las entidades involucradas cumplen las reglas del protocolo, pero esto no es siempre así, y mucho menos si hablamos de transacciones como las de Comercio Electrónico en un entorno como Internet, por ello es necesario recurrir a Terceras Partes de Confianza (TTP) [PAGNIA].

La TTP puede involucrarse en diferente medida en la ejecución del protocolo, pudiendo intervenir en cada paso del protocolo, o bien solo siendo necesaria su actuación cuando alguna entidad involucrada en el intercambio lo requieran. En este último caso, se habla de protocolos de intercambio justo optimistas. Usando este tipo de TTP, se han definido varios protocolos eficientes y optimistas [OFEP] que, en el mejor de los casos, requieren solo tres mensajes para finalizar su ejecución. El protocolo de

Micali [MICALI] y el protocolo FPH [FPH] son otros ejemplos de protocolos de intercambio justo optimistas.

Otras soluciones asumen que hay un TTP adicional que proporciona un medio para verificación de que la información intercambiada contiene el contenido correcto (por ejemplo, utilizando hashes). Tales entidades que pueden denominase hashes de certificación ya existen en los sistemas de *BitTorrent* actuales [TORRENT].

## 2.5 Protocolo OFEPSP+

En [JLHA] se propone un protocolo de intercambio justo optimista, basado en el uso de Políticas de Firma Extendidas. En este protocolo, se define un protocolo principal en el que no interviene TTP, y subprotocolos de recuperación e interrupción de la ejecución con la actuación de una tercera entidad de confianza.

Tanto en el caso del protocolo principal como en el caso de los subprotocolos, la ejecución se basa en el intercambio de evidencias (firmas electrónicas) generadas en base a unas Políticas de Firma determinadas, y en los que las reglas de generación de las evidencias dentro de una transacción se encuentran determinadas por Políticas de Firma Extendidas.

### 2.5.1 Entidades involucradas

- **Origen (O)**, entidad que inicia la transacción. Utiliza dos entornos distintos (OE1, OE2) para generación y validación de las evidencias. En la implementación realizada, corresponde al Origen que accede a la aplicación del Receptor.

- **Receptor (R)**, entidad que obtiene el mensaje y no repudio del Origen. Debe generar y enviar al Origen las evidencias necesarias para que tanto Origen como Receptor no estén en desventaja. A diferencia del Origen, dispone de un entorno único.

- **TTP-SP**, tercera entidad de confianza encargada de emitir y publicar las políticas de firma y políticas de firma extendidas que usan el resto de entidades para crear y validar las evidencias intercambiadas. En la implementación realizada, estos servicios los proporciona el Prestador de Servicios de Certificación (PSC) implementado.

- **TTP**, tercera entidad de confianza que solo interviene si se detecta una ejecución anormal del protocolo principal. En la implementación realizada corresponde a una aplicación a la que pueden acceder el Origen y el Receptor para ejecutar los subprotocolos de interrupción y recuperación respectivamente.

### 2.5.2 Evidencias intercambiadas

- **PNRO1**, evidencia inicial generada por el Origen en su entorno OE1. Sirve para iniciar el intercambio mediante la firma de los datos a intercambiar, y firma de la información que identifica la ejecución que se inicia (identificador de la ejecución).

- **PNRR1**, evidencia generada por el Receptor posteriormente a la validación de PNRO1. Consiste en la firma por parte del Receptor de los datos a intercambiar, y del identificador de la ejecución.

- **PNRO2**, evidencia generada por el Origen en su entorno OE2, posteriormente a la recepción y validación de PNRR1 y PNRO2 en dicho entorno. Consiste en una *conterSignature* de PNRR1.

- **PNRR2**, evidencia generada por el Receptor posteriormente a la validación de PNRO2. Consiste en una *conterSignature* de PNRO2.

- **NRE**, evidencia generada por el Origen en su entorno OE1, posteriormente a la recepción y validación de PNRR2 en dicho entorno. Consiste en una *conterSignature* de PNRR2. La validación correcta de esta evidencia por el Receptor finaliza la ejecución del protocolo. En el caso de no llegar a generarse, el Receptor podrá solicitar a TTP la generación de **NRE-TTP** en la ejecución del subprotocolo de recuperación.

- En ejecuciones del protocolo de interrupción, se intercambian **Peticiones de Interrupción** generadas por el Origen y enviadas al TTP, así como **Evidencias e Interrupción** generadas por el TTP.

### 2.5.3 Protocolo principal

El protocolo principal, descrito en el apartado 9.1.3 de [JLHA], se compone de los siguientes pasos que pueden ejecutarse con la implementación realizada:

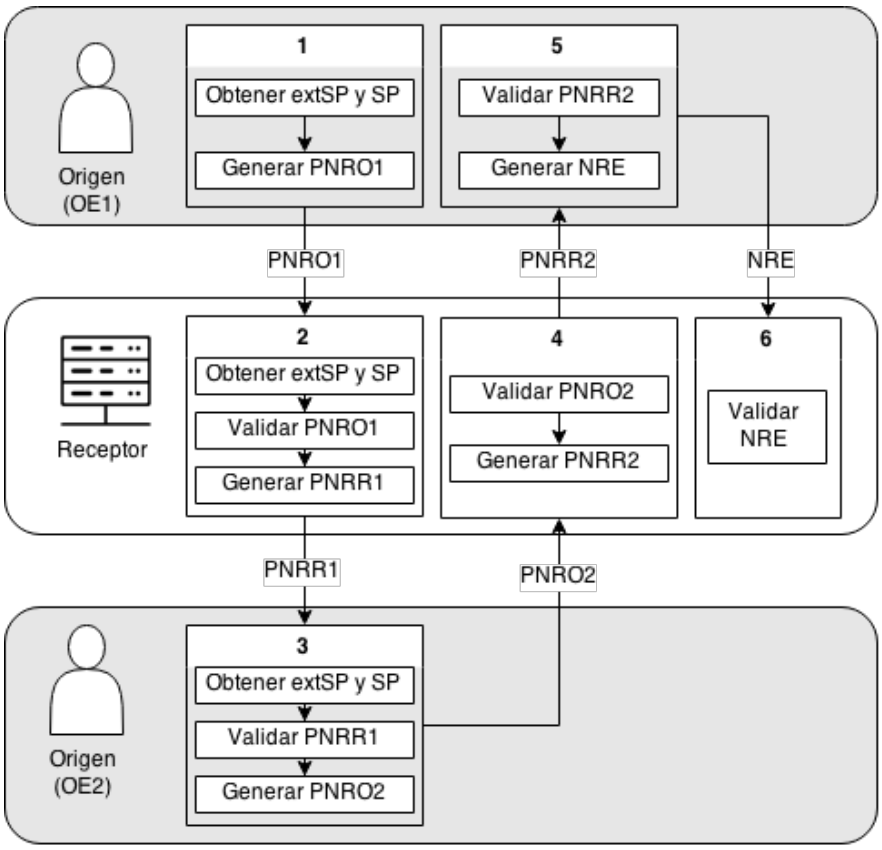



*Figura 3: Ejecución protocolo principal*

1. El Origen, en su entorno inicial OE1, obtiene la política de firma extendida a usar (extSP), así como la política de firma (SP) con las reglas para generar PNRO1. Con esta información genera la evidencia inicial mediante la firma del mensaje a intercambiar y la información necesaria para identificar la ejecución. Una vez generado, el Origen manda PNRO1 al Receptor, iniciando de este modo la ejecución del protocolo.

2. El Receptor, obtiene la extSP y la SP indicadas en PNRO1 y realiza la validación de la firma según las reglas y condiciones de las políticas obtenidas. Una vez validado, genera PNRR1 y lo pone a disposición del Origen en su entorno OE2.

3. El Origen, desde su entorno OE2 realiza la validación de PNRR1, para ello obtiene la extSP y la SP indicadas en PNRR1. La validación de PNRR1 implicará al Origen en OE2 realizar adicionalmente la validación de PNRO1 en OE2. Si la validación es correcta, genera en OE2 la evidencia PNRO2 que envía al Receptor.

4. El Receptor, obtiene y valida la evidencia generada por el Origen en OE2, si la validación es satisfactoria genera PNRR2 realizando una *conterSignature* de PNRO2 y pone a disposición del Origen en OE1 la evidencia generada.

5. El Origen, obtiene en OE1 la evidencia PNRR2, realiza la validación de la misma realizando a su vez la validación de la evidencia PNRO2 que se ha generado en OE2. Si la validación es correcta, generará NRE y enviá dicha evidencia de no repudio al Receptor.

6. El Receptor valida NRE según las condiciones y reglas de las políticas, finalizando la ejecución del protocolo intercambio justo.

### 2.5.4 Subprotocolo de recuperación

El subprotocolo de recuperación permite al Receptor obtener NRE en los casos que se produzca una interrupción de la ejecución o una mala conducta por parte del Origen. Este subprotocolo se ejecutará si el Receptor no obtiene NRE dentro de un intervalo de tiempo especifico. Este subprotocolo, descrito en el apartado 9.1.4 de [JLHA], de forma resumida se compone de los siguientes pasos:

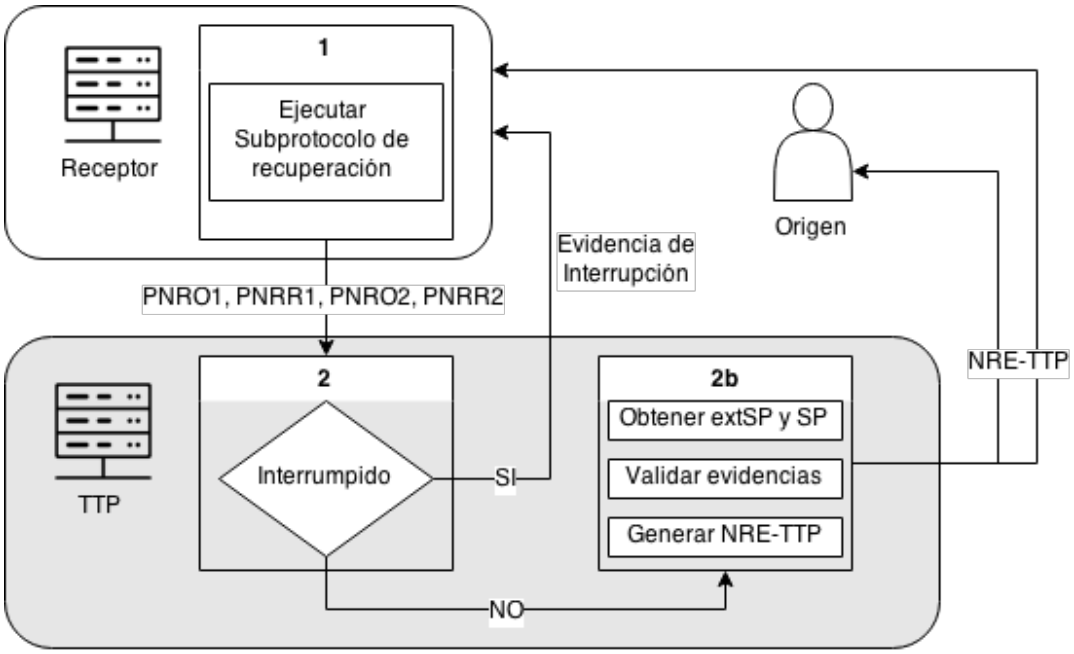



*Figura 4: Ejecución subprotocolo de recuperación*

1. El Receptor puede iniciar la ejecución del subprotocolo de recuperación enviando a TTP las evidencias generadas: PNRO1, PNRR1, PNRO2, PNRR2. Teniendo en cuenta que en casos de intercambio de información confidencial, no se le enviá a TTP los datos intercambiados, sino que el TTP obtiene del Receptor el hash necesario de dicha información para poder realizar la validación de PNRO1 y PNRR1.

2. El TTP tendrá un comportamiento distinto dependiendo de si se solicitó la interrupción de la ejecución.

   2.a.) Si se ejecutó la interrupción del protocolo, TTP enviará la evidencia de interrupción al Receptor.

   2.b.) En caso contrario, una vez validadas las evidencias obtenidas, genera NRE-TTP, que pondrá a disposición del Receptor y del Origen.

### 2.5.5 Subprotocolo de interrupción

El subprotocolo de interrupción permite al Origen abortar la ejecución del protocolo si sospecha de un comportamiento malicioso por parte del Receptor, si se ha producido un error durante la ejecución del protocolo o, más importante, si detecta una transacción fraudulenta en las fases de validación en cualquiera de sus entornos. El subprotocolo de interrupción puede ser ejecutado por el Origen en cualquier paso del protocolo principal y desde cualquiera de sus entornos. Este subprotocolo, descrito en el apartado 9.1.5 de [JLHA], de forma resumida se compone de los siguientes pasos:

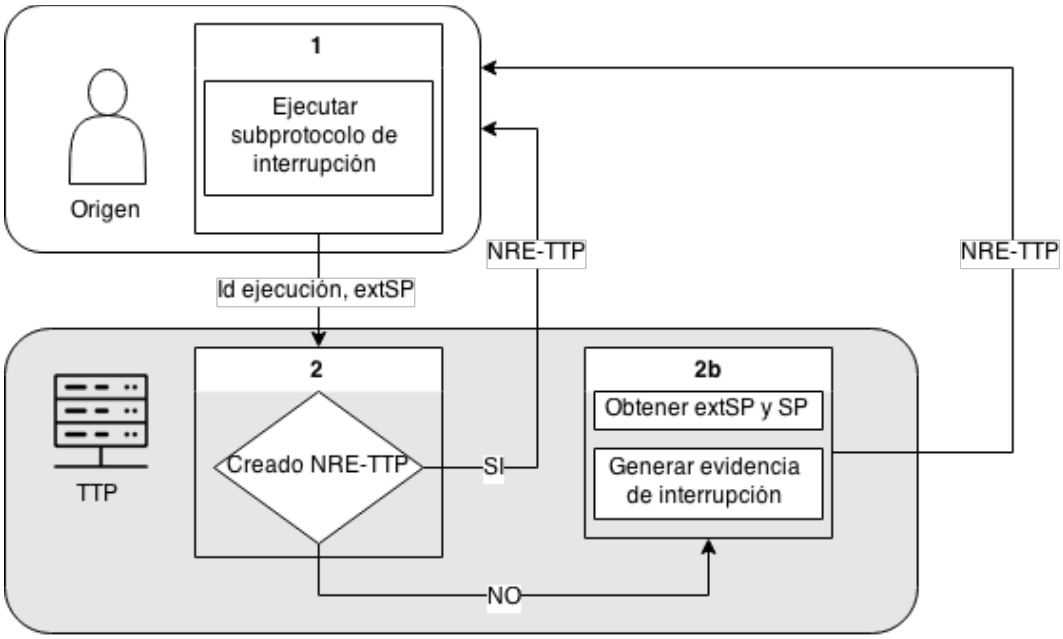



*Figura 5: Ejecución subprotocolo de interrupción*

1. El Origen puede iniciar la ejecución del subprotocolo de interrupción enviando a TTP una petición de interrupción firmada, esta petición incluirá el identificación de la ejecución a interrumpir.

2. El TTP tendrá un comportamiento distinto dependiendo de si se ejecutó el subprotocolo de recuperación.

2.a.) Si se ejecutó la el subprotocolo de recuperación, TTP enviará la evidencia NRE-TTP al Origen.

2.b.) En caso contrario, una vez validada la firma de la petición de interrupción, el TTP generará la evidencia de interrupción mediante la realización de una *counterSignature* de la petición, dejando constancia de la interrupción del protocolo principal. El TTP pone a disposición del Origen dicha evidencia.

# Capítulo 3. Análisis Funcional e Implementación de OFEPSP+

## 3.1 Alcance de la implementación

El objetivo de la implementación de las entidades involucradas en la ejecución del protocolo, es el de simular tres escenarios que cubran los casos de uso descritos en el apartado 10.3.3 de [JLHA] :

- Una ejecución de OFEPSP+ donde el Origen y el Receptor se comunican entre sí de una manera normal, sin interactuar con el TTP.

- Una ejecución de OFEPSP+ donde se detecta un comportamiento malicioso por parte del Receptor. En este caso, el Receptor no envía PNRR2 al Origen o se detecta en alguno de los entornos del Origen un comportamiento malicioso por parte del Receptor, en este caso se ejecuta el subprotocolo de interrupción.

- Una ejecución de OFEPSP+ donde se simula un error de comunicación de red que obliga la ejecución de los subprotocolos. En particular, el Receptor envía PNRR2 al Origen, pero se produce un fallo de comunicación que no permite al Origen generar y/o enviar NRE. Posteriormente, y debido a que el Receptor no recibe NRE, se ejecuta el subprotocolo recuperación, obteniéndose NRE-TTP del TTP.

Para cubrir estos casos de uso, es necesario implementar cada una de las identidades involucradas en el protocolo, así como todos los servicios de certificación; el alcance de la implementación de las distintas entidades se acotará, centrándose en cumplir los requisitos para la ejecución del protocolo principal y ambos subprotocolos.

Se implementará el protocolo con un modelo B2C en el que, para la ejecución del protocolo principal, una persona física que representa la entidad Origen del protocolo accede a una aplicación Web que pertenece a la entidad Receptor, mientras que para la ejecución de los subprotocolos, una persona física que representa la entidad Origen o la entidad Receptor (dependiendo del subprotocolo a ejecutar) accede a una aplicación Web que representa a la entidad TTP del protocolo.

Para que se encuentren disponibles los servicios de certificación necesarios en la ejecución, en primer lugar se implementará un Prestador de Servicios de Certificación (PSC) que cubra la funcionalidad necesaria para generar y publicar SP/extSP, así como para validar el estado de revocación de certificados. La implementación se limitará a la generación de SP con unas reglas comunes configurables y reglas de los niveles de compromiso utilizados en OFEPSP+. En cuanto a la generación de extSP, el PSC permitirá generar extSP con los Árboles de Solución definidos para OFEPSP+, pudiéndose determinar para cada nodo de dichos árboles la SP a usar y las condiciones de tiempo aceptable de generación respecto al resto de nodos.

Respecto a los servicios de emisión y validación de certificados, se generará un juego de pruebas de certificados para las distintas entidades, y el PSC publicará una CRL con algunos de ellos revocados. Adicionalmente se implementará un servicio OCSPResponder que compruebe el estado de revocación consultando la CRL y además establezca otros certificados revocados con fecha de revocación posterior a la publicación de la CRL.

La implementación permitirá definir en las SP otros tipos de certificados de otros PSC, pero únicamente se podrán utilizar en la ejecución del protocolo implementado si se encuentran en estructuras PKCS#12, y teniendo en cuenta que no se podrá validar el estado de revocación de dichos certificados en caso de que la SP obligue a ello. Las distintas entidades podrán decidir a qué PSC llamar en las validaciones de certificados indicando el nombre de máquina y puerto del PSC, pero para ello dicho PSC deberá tener publicados los servicios de validación en unas rutas específicas.

Tanto el formato de SP como de extSP y de firmas será XML. En el caso particular de las firmas, el alcance de la implementación en cuanto a su generación y validación, se basará en estructuras XAdES v1.3.2 con perfil -EPES, implementando las modificaciones necesarias a XAdES para cumplir con los requisitos de las firmas en cuanto al uso de Políticas de Firma Extendidas y ejecución de OFEPSP+.

No se contempla la implementación de sistemas de resolución de disputas, en los que una entidad independiente a la que podemos denominar Juez, se encargue de resolver disputas con las evidencias obtenidas de las distintas entidades.

## 3.2 Arquitectura de la solución

La implementación realizada se divide en distintos módulos dependiendo de las necesidades funcionales y de las entidades a simular en la ejecución del protocolo. Los módulos implementados son los los siguientes:

- ***ofepspplus_signature:*** módulo de firma e interpretación de políticas, proporciona la funcionalidad necesaria para la generación y validación de firmas electrónicas XAdES con perfil -EPES, incluida la generación y validación de las evidencias necesarias en el protocolo mediante el uso de extSP. Adicionalmente implementa la funcionalidad necesaria para la creación e interpretación por parte de las distintas entidades de SP y extSP en formato XML.

- ***ofepspplus_psc:*** Prestador de Servicios de Certificación, se trata de una aplicación Web con la funcionalidad necesaria para la emisión y publicación de las SP y extSP a utilizar por el resto de entidades. También se encarga de publicar los servicios de certificación básicos que son necesarios para la ejecución del protocolo: emisión de certificados reconocidos, y publicación de servicios de validación de dichos certificados, CRL y OCSPResponder.

- ***ofepspplus_buyerApplet****:* este módulo corresponde al componente software que usará el Origen en su entorno local para generar y validar las evidencias necesarias en sus entornos OE1 y OE2, también proporciona al Origen la funcionalidad necesaria para la obtención e interpretación de SP y extSP. El Origen podrá obtener este componente software desde las aplicaciones Web implementadas por los módulos *ofepspplus_ttp* y *ofepspplus_vendor***.**

- ***ofepspplus_vendor:*** aplicación Web del Receptor, a la que el Origen accede para iniciar la ejecución del protocolo de intercambio y para interactuar con el Receptor en la ejecución del protocolo principal. Implementa la funcionalidad de la entidad Receptor del protocolo en la ejecución del protocolo principal, realizando la validación y generación de evidencias de dicha entidad. A su vez gestiona los estados de las ejecuciones en curso y finalizadas del protocolo. Proporciona a la entidad Origen el componente de software para generación y validación de evidencias (*ofepspplus_buyerApplet*).

- ***ofepspplus_ttp:*** se trata de una aplicación Web del TTP que publica los servicios necesarios para cubrir la funcionalidad de la TTP. Al tratarse de un protocolo de intercambio justo optimista, estos servicios cubren las necesidades para ejecutar el subprotocolo de recuperación, así como la funcionalidad necesaria para la interrupción del protocolo; el usuario de esta aplicación web corresponde a la entidad Origen o entidad Receptor del protocolo dependiendo del subprotocolo que se ejecute. En ejecuciones del subprotocolo de interrupción proporciona al usuario (Origen) el componente de software para generación de peticiones de interrupción firmadas (*ofepspplus_buyerApplet*).

Estos módulos comparten utilidades para la generación y validación de evidencias y a su vez pueden comunicarse entre ellos para el envío y recepción de evidencias. Estos componentes se resumen en

tres aplicaciones Web independientes (*ofepspplus_psc*, *ofepspplus_vendor* y *ofepspplus_ttp*) que se comunican entre ellas y con los usuarios en los procesos de intercambio.

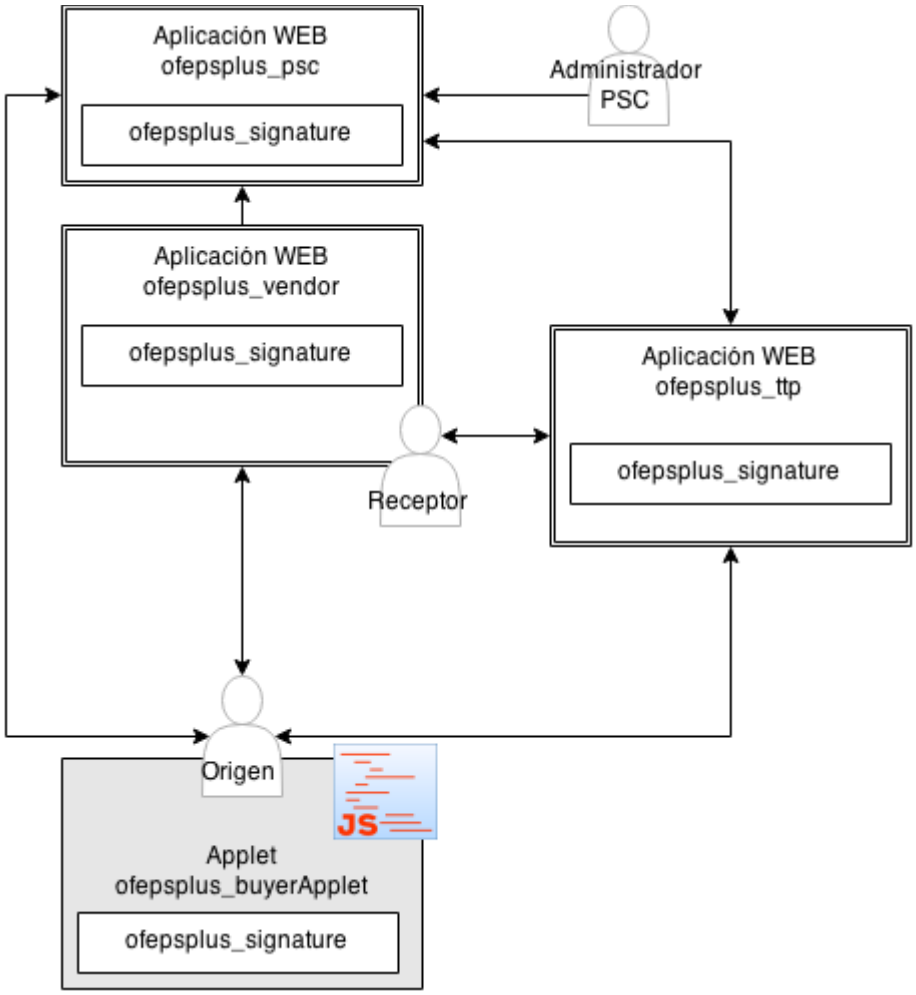



*Figura 6: Arquitectura de la implementación*

Los usuarios de cada una de las aplicaciones Web son los siguientes:

- **Administrador PSC**, accede a la aplicación Web del PSC para usar los servicios de creación y publicación de SP y extSP.
- Un usuario que representa la entidad **Origen** del protocolo. Puede acceder a la aplicación Web *ofepsplus_vendor* para la ejecución del protocolo principal. A su vez accede como usuario de la aplicación Web *ofeplsplus_ttp* para la ejecución del subprotocolo de interrupción. La entidad **Origen** también es usuaria de los servicios de la aplicación Web *ofeplsplus_psc* usando el componente software *ofeplsplus_buyerApplet* como cliente de servicios de certificación. El usuario que representa la entidad **Origen** se comunica con el resto de entidades (Receptor y TTP) ejecutando en su entorno local los métodos del Applet y

las funciones javaScript publicadas en los formularios que obtiene en su navegador web al acceder a las distintas aplicaciones Web.

- Un usuario que representa la entidad **Receptor** del protocolo, puede acceder a la aplicación Web *ofeplsplus_ttp* para la ejecución del subprotocolo de recuperación.
- Las aplicaciones Web ***ofepsplus_vendor***, ***ofeplsplus_ttp*** son clientes de los servicios del módulo Web *ofeplsplus_psc* para la consulta de SP/extSP y de estados de revocación de certificados OCSP/CRL.

## 3.3 Tecnología

La implementación y ejecución del protocolo se realiza en un entorno JEE con las siguientes herramientas de desarrollo y despliegue:

- IDE de Desarrollo *Eclipse Java EE IDE for Web Developers, version: Indigo Service*
- *Java Development Kit* versión 1.7.0 update 45.
- *Maven2,* para la generación de binarios y desplegables.
- *Tomcat7,* como servidor de aplicaciones.
- Proveedor criptográfico *BouncyCastle*
- *JSF2/PrimefacesFaces5* para implementar la lógica de negocio y capa de presentación de los módulos Web.
- *SQLite3*, se usará como gestor de base de datos que será accesible por los módulos del proyecto a través de *JPA/Hibernate*.
- *XCA*, se usará para la emisión de certificados reconocidos y generación de CRLs por parte del Prestador de Servicios de Certificación.

## 3.4 *ofepspplus_signature*, Módulo de firma e interpretación de políticas

### 3.4.1 Análisis funcional

Este módulo funcional abarca las necesidades de las distintas entidades involucradas en la ejecución del protocolo en cuanto a la generación y validación de firmas electrónicas, así como uso de SP y extSP. Para ello cubrirá la funcionalidad descrita en los siguientes apartados.

#### *3.4.1.1Generación e interpretación de SP y extSP*

El prestador de servicios implementado debe emitir las SP y extSP que puedan ser interpretadas y usadas por las distintas entidades involucradas en el protocolo (Origen, Receptor y TTP). Para ello, tanto las SP como las extSP se publicarán en formato XML.

El módulo *ofepspplus_signature* proporciona la funcionalidad necesaria para la generación e interpretación de SP en Formato XML [TR102038], así como para permitir su interpretación. También proporciona la misma funcionalidad para la creación e interpretación de extSP en formato XML [JLHA].

#### *3.4.1.2Generación y validación de evidencias*

El módulo *ofepspplus_signature* proporciona la funcionalidad necesaria para la creación y validación de las evidencias necesarias en la ejecución del protocolo, esta generación de evidencias tiene que basarse en las reglas existentes en las SP y en la información de las extSP que se vayan a usar en cada transacción. Para ello implementará la generación y validación de las evidencias ya descritas en el *apartado 2.5.2*.

Para permitir la generación de estas evidencias, este módulo debe proporcionar la generación de jerarquías de firmas tanto simples, como firmas en paralelo y *conterSignatures*. Y en todos casos basándose en las reglas que se obtengan de las SP; y teniendo en cuenta en los procesos de validación los requisitos de las firmas dentro de la transacción según las extSP.

La relación entre las firmas generadas en la generación de evidencias y por tanto la jerarquía de las firmas a generar, se encuentra definida en los árboles de solución de extSP. Para el caso de los árboles de solución de OFEPSP+ cada una de las evidencias se componen de firmas simples, en paralelo y *CounterSignatures* siendo necesario generar en cada caso las siguientes firmas:

- **Evidencia PNRO1**, se genera una firma simple de los Datos a intercambiar y la información de identificación de la ejecución.

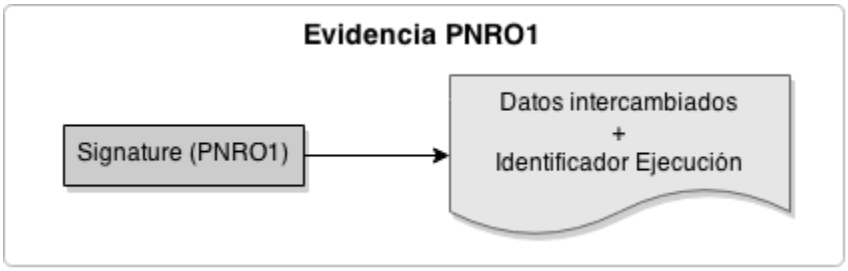



*Figura 7: Jerarquía de firmas al generar PNRO1*

- **Evidencia PNRR1**, se incluye una firma en paralelo de los Datos a Intercambiar y de la información de identificación de la ejecución.

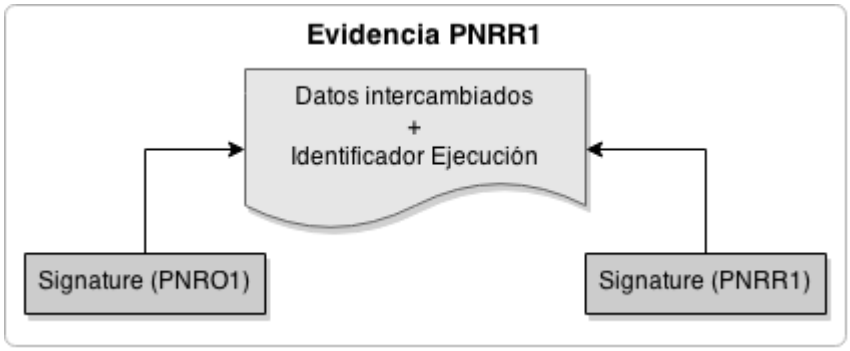
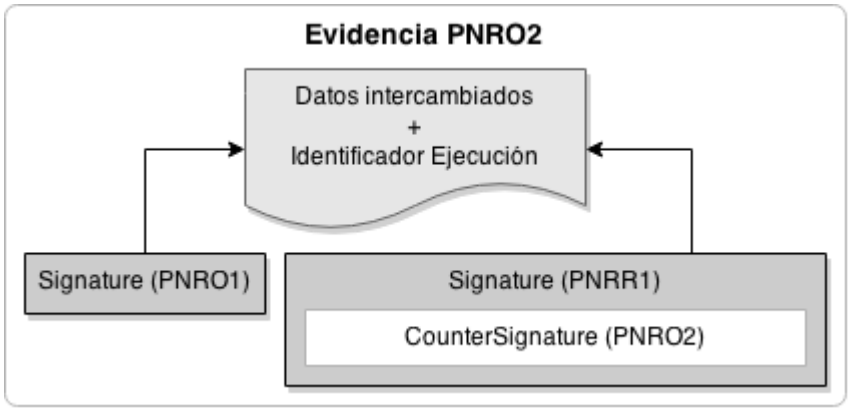



*Figura 8: Jerarquía de firmas al generar PNRR1*

- **Evidencia PNRO2**, se incluye un elemento *CounterSignature* de la firma generada al crear PNRR1.




- *Figura 9: Jerarquía de firmas al generar PNRO2*

- **Evidencia PNRR2**, se incluye un elemento *CounterSignature* de la firma generada al crear PNRO2.

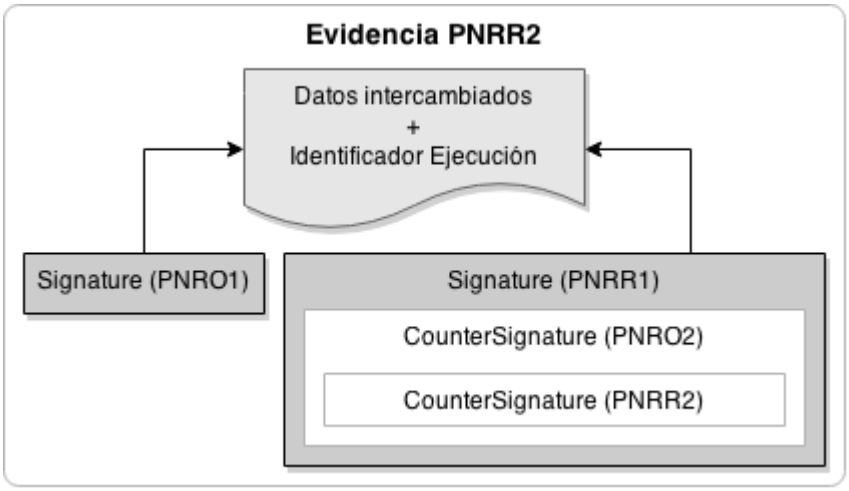



*Figura 10: Jerarquía de firmas al generar PNRR2*

- **Evidencia NRE / NRE-TTP**, se incluye un elemento *CounterSignature* de la firma generada al crear PNRR2.

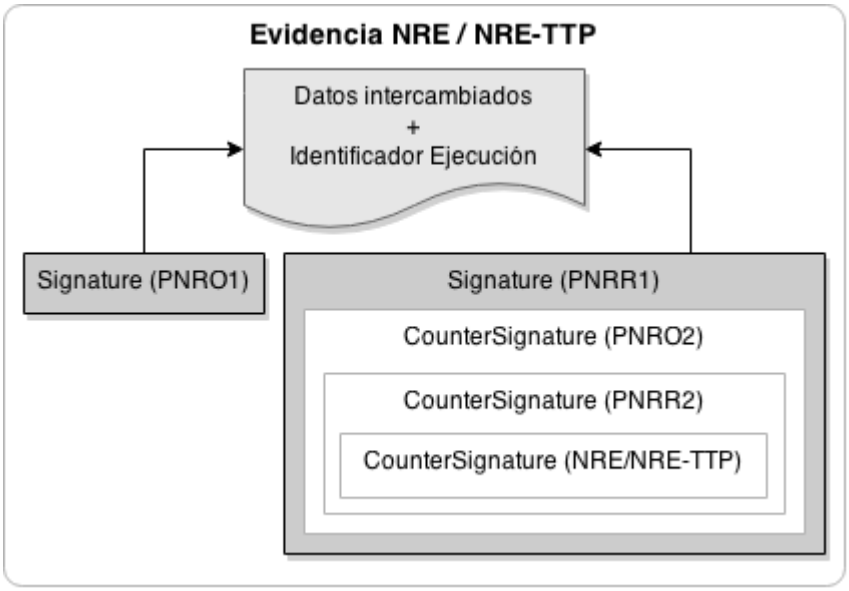



*Figura 11: Jerarquía de firmas al generar NRE/NRE-TTP*

Adicionalmente debe proporcionar la funcionalidad necesaria para generar solicitudes de interrupción, cuya generación en este caso no se basa en SP/extSP. Para ello debe permitir generar la firma simple de peticiones de interrupción, y la generación de evidencias de interrupción incluyendo un elemento *CounterSignature* de la firma de la petición.

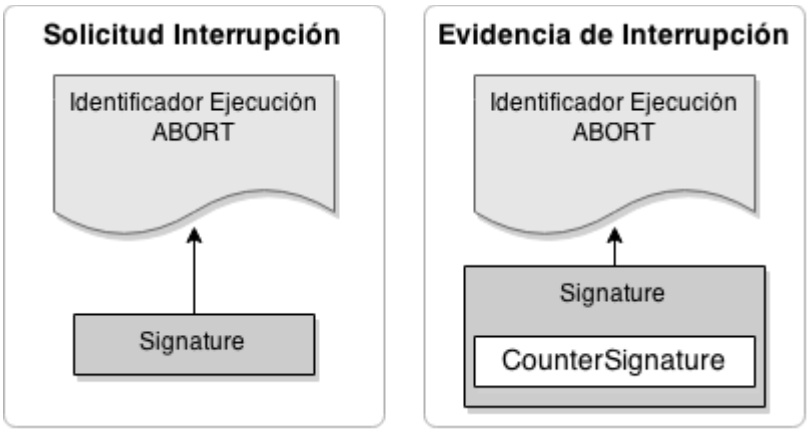



*Figura 12: Jerarquía de firmas en la ejecución de subprotocolo de interrupción*

#### *3.4.1.3Funciones comunes de servicios de criptografía y servicios de certificación*

Se debe proporcionar la funcionalidad necesaria por todas las entidades para el acceso a los servicios de certificación que se requieran, como por ejemplo permitir realizar peticiones para la consulta de SP y extSP, realización de llamadas a OCSPResponder, o realizar validación de certificados mediante la consulta de CRLs publicadas.

Adicionalmente, este componente incluirá el resto de funcionalidad común, como el uso de almacenes de certificados tipo PKCS#12, y acceso a funcionalidad de los proveedores criptográficos.

### 3.4.2 Implementación

#### *3.4.2.1Generación e interpretación de SP y extSP*

Para implementar el módulo se utiliza *Java Architecture for XML Binding 2.0* [JSR222]. JAXB proporciona dos características principales: la capacidad de serializar las referencias de objetos Java a XML y la inversa, es decir, deserializar XML en objetos Java. En otras palabras, JAXB nos permite generar las clases necesarias para almacenar y recuperar datos de las políticas en memoria partiendo su definición en formato XML, sin la necesidad de implementar un conjunto específico de rutinas de carga y guardado de XML para la estructura de clases.

Esto nos permite que a partir de los XSD que definen la estructura en XML de las SP y extSP, se generen las clases Java necesarias para la creación y parseo de dichos XML a partir de POJOs. Obteniéndose la estructura de clases a utilizar por los distintos módulos que requieran la funcionalidad de creación o parseo de estos XML.

Las clases generadas con JAXB en la implementación del protocolo se crean en dos paquetes, cada uno de ellos contiene las clases necesarias para realizar el mapeo de cada uno de los espacios de nombres usados en la definición de SP y extSP respectivamente:

| **PACKAGE** | **NAMESPACE** |
|---|---|
| es.uc3m.ofepsp.api_signature.jaxb.signaturepolicy | http://uri.etsi.org/2038/v1.1.1# |
| es.uc3m.ofepsp.api_signature.jaxb.extsignaturepolicy | http://jlopez.thesis.uc3m.es/ETS-ExtendedElectronicSignaturePolicies-97Syntax |

*Tabla 3: Espacios de nombre generados con JAXB*

De forma particular a continuación se muestra el diagrama con las clases principales creadas para trabajar con las SP:

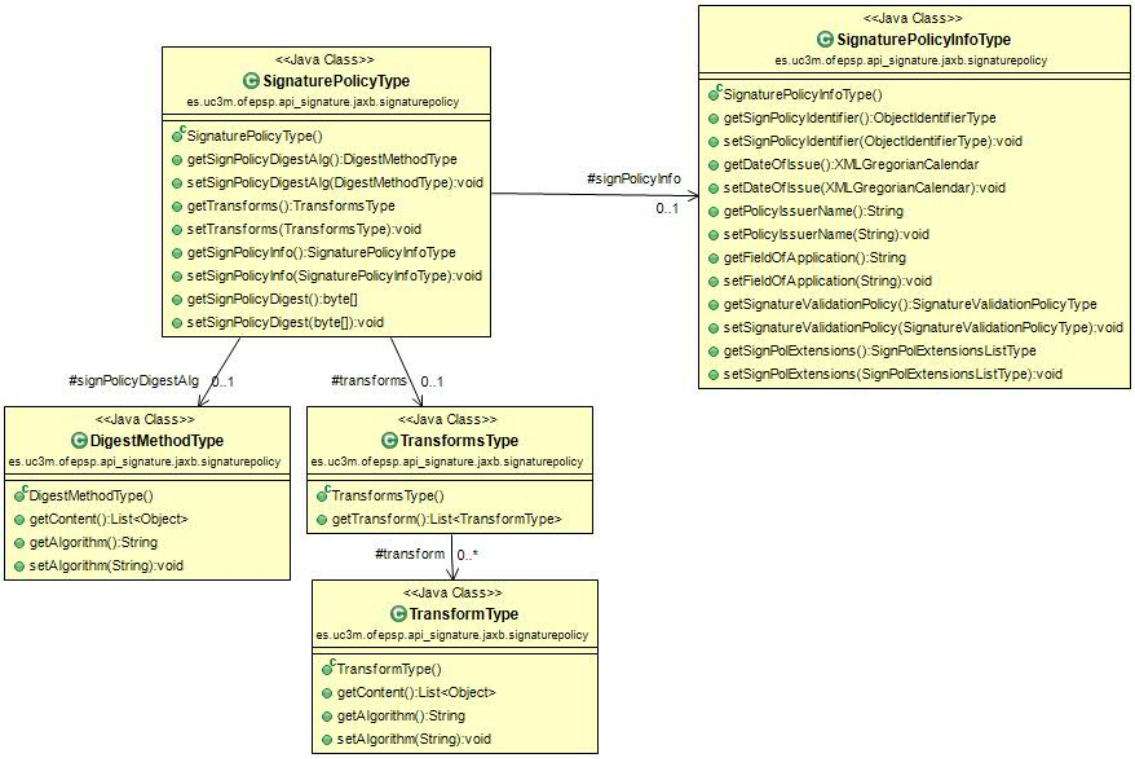



*Figura 13: Clases principales para la generación e interpretación de Políticas de Firma*

La clase *SignaturePolicyType* representa el contenido de una SP, mientras que la clase *SignaturePolicyInfoType* implementa el contenido de la Política de Validación usando el resto de clases creadas con JAXB en el mismo paquete, de tal forma que estas clases, junto con el resto existente en el mismo paquete, representan todos los tipos definidos en [TR102038] y sus dependencias.

Del mismo modo, a continuación se muestra el diagrama con las clases principales generadas que se usarán cuando se necesite trabajar con extSP:

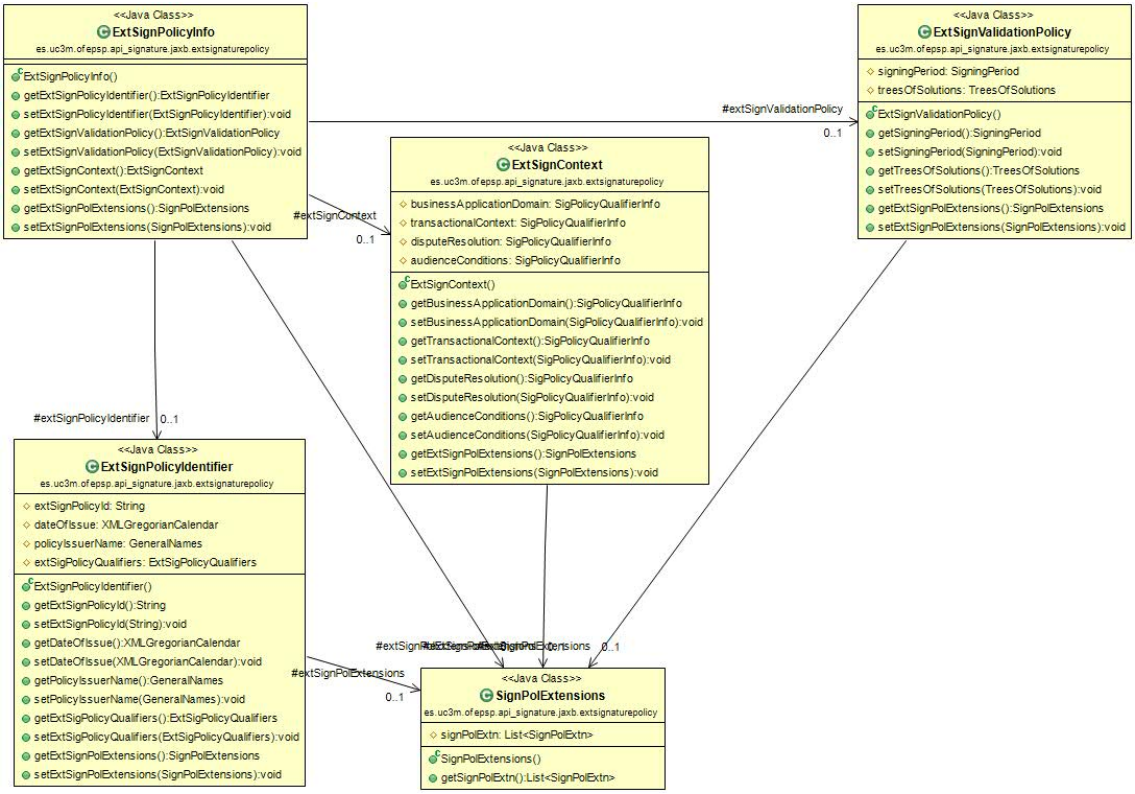



*Figura 14: Clases principales para la generación e interpretación de Políticas de Firma Extendidas*

La clase *ExtSignPolicyInfo* representa el contenido de una extSP. La clase *ExtSignValidationPolicy* implementa el contenido de la Política de Validación, especialmente los árboles de solución. Mientras que la clase *ExtSignContext* encapsula la información del contexto de aplicación de la política. Estas clases, junto al resto resto de clases creadas con JAXB en el mismo paquete, representan todos los tipos existentes en la definición de las extSP en formatos XML [JLHA] y sus dependencias.

### *3.4.2.2Generación de firmas electrónicas*

El módulo *ofepspplus_signature* permite generar firmas en formato XML, con perfiles XAdES EPES, y en el se encuentra la implementación necesaria para la validación de dichas firmas basándose en las reglas de validación definidas en las SP y en los requisitos de las firmas dentro de una determinada transacción existentes en las extSP.

Se implementa para ello un motor de generación de firmas XAdES, partiendo de su versión 1.3.2 [XADES132], pero siendo necesario modificar las estructuras de firma XML para que se adapten a las necesidades del protocolo. Para ello, las firmas a generar se basan en el [XADES132] pero se añade una nueva propiedad de firma que se añadirá en las firmas electrónicas durante de la generación de las

evidencias en la ejecución del protocolo. Esta nueva propiedad se añade como un nuevo elemento incluido en las propiedades firmadas (*SignedSignaturePropertiesType*): *ExtSignaturePolicyIdentifier* de tipo *SignedSignaturePropertiesType* (tipo ya definido en [XADES132]). Adicionalmente, se modifica la propiedad *SignatureProductionPlace* añadiendo el elemento *Envioronment*, que permite añadir como propiedad firmada la identificación del entorno en el que se ha generado la evidencia.

En resumen, las firmas generadas durante la ejecución del protocolo serán estructuras XAdES v.1.3.2 con las siguientes modificaciones que permiten cumplir los requisitos de las evidencias intercambiadas:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://uri.etsi.org/01903/v1.3.2EXT#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://uri.etsi.org/01903/v1.3.2EXT#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" elementFormDefault="qualified">
[...]
<xsd:complexType name="SignedSignaturePropertiesType">
<xsd:sequence>
<xsd:element name="SigningTime" type="xsd:dateTime" minOccurs="0"/>
<xsd:element name="SigningCertificate" type="CertIDListType" minOccurs="0"/>
<xsd:element name="SignaturePolicyIdentifier"
type="SignaturePolicyIdentifierType" minOccurs="0"/>
<xsd:element name="ExtSignaturePolicyIdentifier"
type="SignaturePolicyIdentifierType" minOccurs="0"/>
<xsd:element name="SignatureProductionPlace" type="SignatureProductionPlaceType"
minOccurs="0"/>
<xsd:element name="SignerRole" type="SignerRoleType" minOccurs="0"/>
</xsd:sequence>
<xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>
[...]
<xsd:complexType name="SignatureProductionPlaceType">
<xsd:sequence>
<xsd:element name="City" type="xsd:string" minOccurs="0"/>
<xsd:element name="StateOrProvince" type="xsd:string" minOccurs="0"/>
<xsd:element name="PostalCode" type="xsd:string" minOccurs="0"/>
<xsd:element name="CountryName" type="xsd:string" minOccurs="0"/>
<xsd:element name="Environment" type="xsd:string" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
[...]
</xsd:schema>
```

Para la implementación de la generación de firmas XAdES se parte del proyecto jXAdES [JXADES], modificando la lógica de generación para adaptar las modificaciones necesarias a la versión 1.3.2 de

XAdES, el código reutilizado y modificado del proyecto jXAdES se incluye en el paquete: *net.java.xades* del módulo *ofepspplus_signature.*

Además de adaptar la implementación para crear estructuras acordes a las modificaciones realizadas sobre [XADES132], se han realizando las implementaciones necesarias para que se generen ciertos elementos en las estructuras XAdES que no se encuentran contemplados en la implementación jXAdES. Estas mejoras sobre el código de partida han sido básicamente la implementación de creación de elementos *CommitmentTypeIdentifier* en las estructuras XAdES y la posibilidad de incorporar la propiedad firmada *SignatureProductionPlace*.

Las jerarquías de firmas que se pueden generar con jXAdES inicialmente cubren los requisitos de las jerarquías a generar durante la ejecución del protocolo, permite generar tanto firmas simples, como *counterSignatures* y firmas en paralelo. Los métodos para generación de *counterSignatures* de jXAdES generan una firma de la primera firma existente en la estructura XML de entrada, esto es un problema a la hora de generar las evidencias del protocolo donde es necesario firmar *counterSignatures* de *counterSignatures*.

Por ello se ha modificado en el comportamiento de los métodos de generación de *counterSignature* implementados en jXAdES. Se ha realizado la implementación necesaria para permitir indicar el nivel de compromiso a contrafirmar en procesos de generación de *counterSignatures*, de tal modo que se añade un elemento *CounterSignature* a la firma que tenga el nivel de compromiso indicado.

| **Evidencia a Generar** | **Nivel de compromiso a contrafirmar** |
|---|---|
| PNRO1 | *No se genera una CounterSignature, se firman los datos e identificador de ejecución* |
| PNRR1 | *No se genera una CounterSignature, se realiza una firma en paralelo de los datos e identificador de ejecución* |
| PNRO2 | http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt1 |
| PNRR2 | http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin2 |
| NRE / NRE-TTP | http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt2 |

*Tabla 4: Niveles de compromiso que se indican a contrafirmar en la generación de evidencias*

Dentro del módulo *ofepspplus_signature* se implementan una serie de clases que permitirán usar el core de firma creado a partir de jXAdES. Estas clases, principalmente son las que se indican en el diagrama de la *Figura 15*.

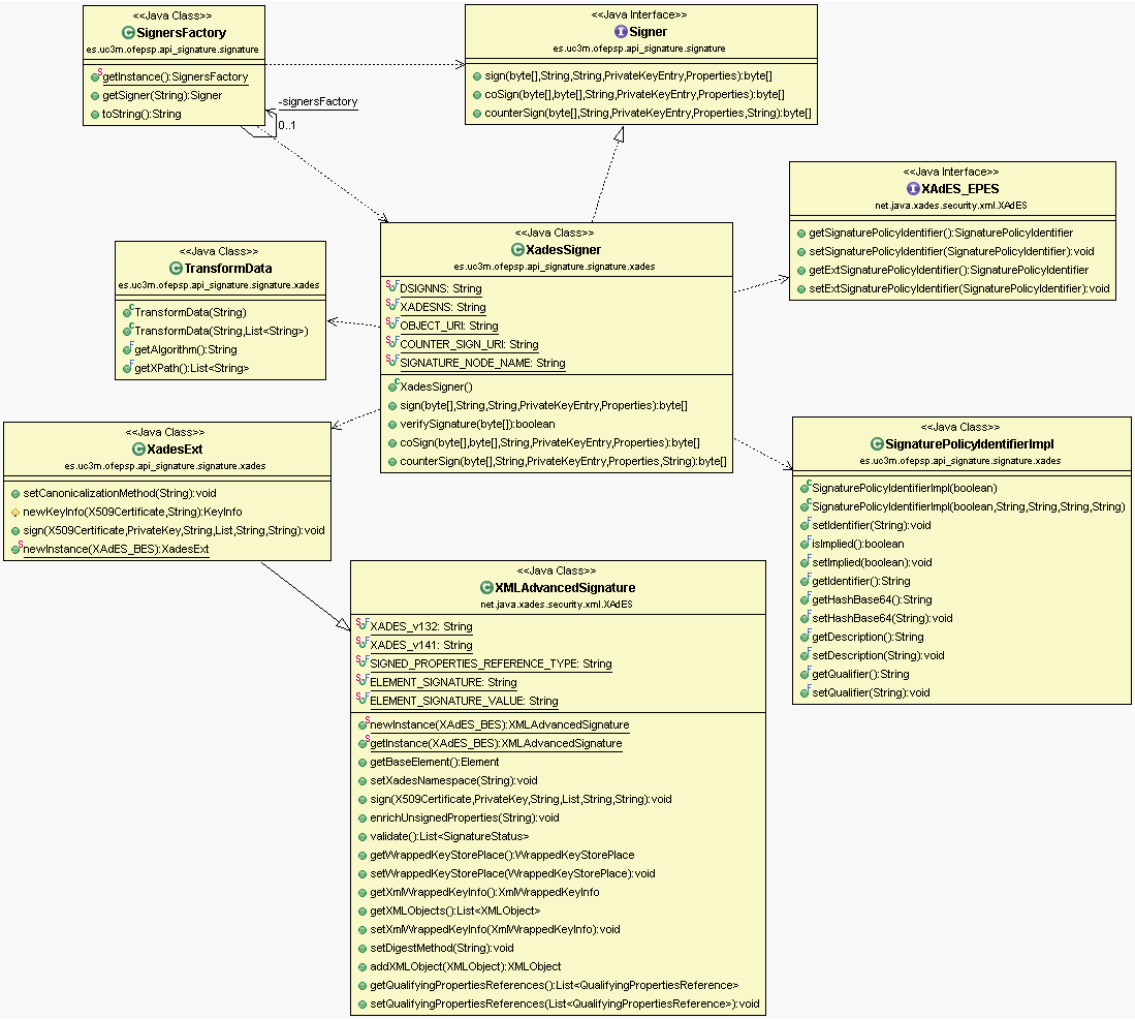



*Figura 15: Clases principales para la generación de firmas electrónicas*

La clase *XadesSigner* proporciona la creación de estructuras XAdES mediante el uso de las clases de *net.java.xades*. Para cumplir con las necesidades en la creación de evidencias del protocolo, *XadesSigner* proporciona funcionalidad para generar firmas simples, firmas en paralelo y firmas en cascada, implementando para ello la interfaz *Signer*. En todos los casos, la generación de estructuras XAdES se realizan teniendo en cuenta las propiedades que se indiquen a incluir en la firma, como son el entorno de generación, nivel de compromiso, política de firma, política de firma extendida, rol, etc.

Una vez implementado el núcleo de generación de firmas XAdES, dentro del paquete *es.uc3m.ofepsp.api_signature.actions* se implementan las clases con las acciones para la generación y validación de evidencias. Estas acciones se encargarán de obtener las reglas de generación/valdiación de las SP y extSP, y con el uso de *XadesSigner* y clases de utilidad existentes generar y validar las evidencias acordes a dichas reglas. Las acciones implementadas deben poder ejecutarse desde aplicaciones servidor (Receptor, TTP) así como aplicaciones cliente (Origen), para ello las acciones implementan la interzaf *java.security.PrivilegedExceptionAction,* permitiéndose su ejecución con

privilegios aunque se trate de una aplicación cliente ejecutada en el contexto de un navegador (Applet). En los siguientes apartados se muestra el diagrama de clases existentes en *ofepspplus_signature* que son usadas por el resto de módulos implementados.

### *3.4.2.3Generación de Evidencias*

Durante la ejecución del protocolo, ante la necesidad de generar una evidencia, las entidades deberán instanciar una de las acciones de creación con los parámetros que correspondan, y ejecutar la acción instanciada para generar la firma electrónica que corresponda.

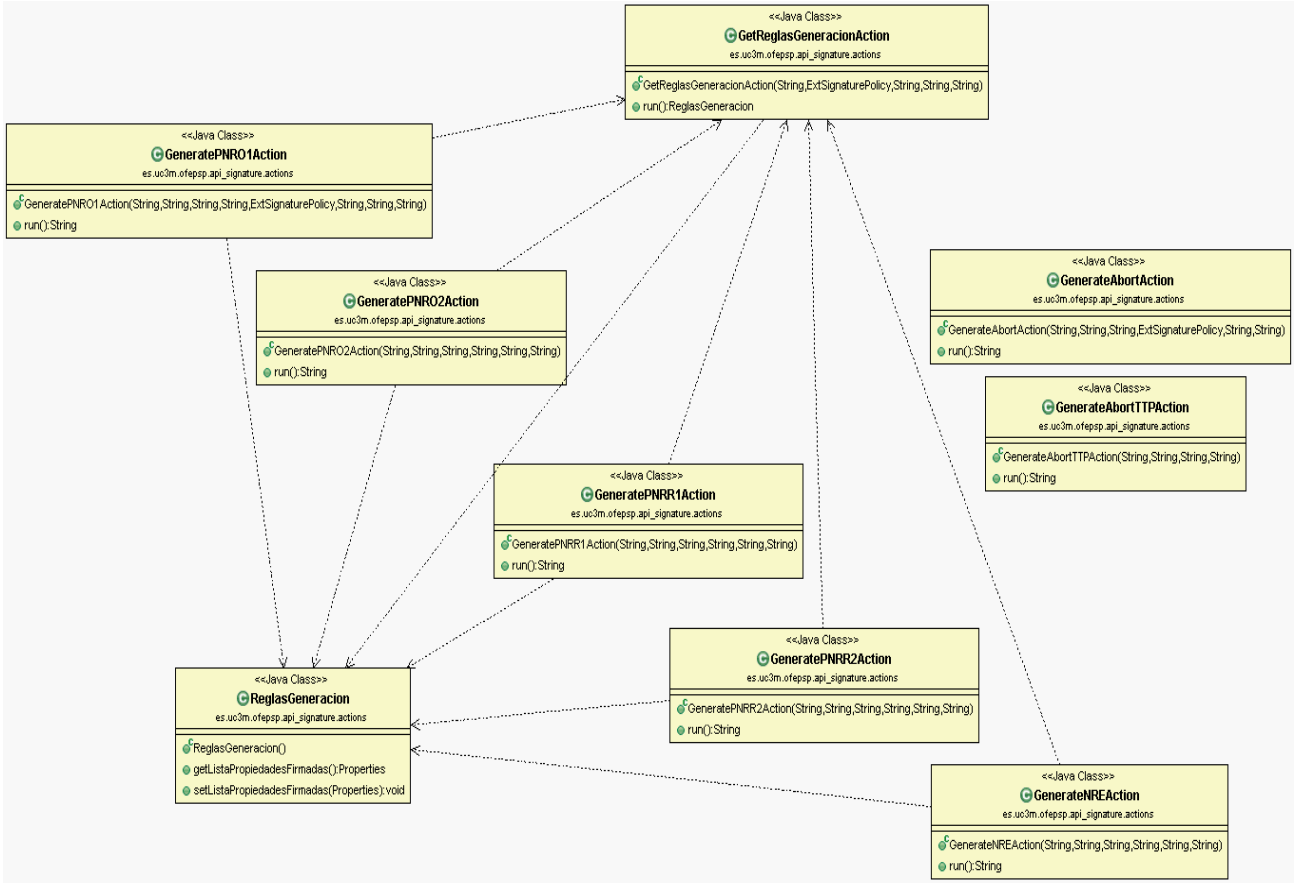



*Figura 16: Diagrama de clases que implementan las acciones de generación de evidencias*

### Acciones para generación de evidencias de protocolo principal

Las acciones implementadas para la generación de evidencias usan la acción implementada en *GetReglasGeneracionAction* para obtener las reglas de generación. Se permite instanciar esta acción indicando el contenido de la extSP a utilizar, el nivel de compromiso del que obtener las reglas, el rol que va a crear la evidencia, la URL donde se encuentra el prestador de servicios de certificación que se usará para obtener la SP, y la identificación del entorno en el que se generará la evidencia. Al ejecutar esta acción se obtiene una instancia de la clase de *ReglasGeneracion* con las reglas a usar en la generación de la firma.

Las reglas de generación obtenidas se componen del listado de propiedades obligatorias a incluir en la firma para el nivel de compromiso indicado, que a su vez se obtienen de la lista de propiedades establecidas como obligatorias en la SP que haya de usarse según la información de la extSP que se haya indicado al obtener las reglas.

Esta lista de propiedades puede incluir cualquier propiedad definida en [XADES132] y adicionalmente la propiedad *ExtSignaturePolicyIdentifier*, pero dependiendo de la propiedad el motor de generación de firmas la incluirá o no dependiendo del tipo, esto permite realizar un gran abanico de pruebas, teniendo en cuenta que en las firmas generadas se incluirán las propiedades o no según la *Tabla 5*.

| Propiedad | Obligada Por la Política | Incluida en la firma |
|---|---|---|
| SigningTime | SI | Siempre se incluye, aunque no sea obligatorio por la política de firma |
| | NO | |
| SigningCertificate | SI | Siempre se incluye, aunque no sea obligatorio por la política de firma |
| | NO | |
| SignaturePolicyIdentifier | SI | SI |
| | NO | NO |
| ExtSignaturePolicyIdentifier | SI | SI |
| | NO | NO |
| SignatureProductionPlace | SI | SI |
| | NO | NO |
| SignerRole | SI | SI |
| | NO | NO |
| DataObjectFormat | SI | Siempre se incluye, aunque no sea obligatorio por la política de firma |
| | NO | |
| CommitmentTypeIndication | SI | SI |
| | NO | NO |
| AllDataObjectsTimeStamp | SI | Nunca se incluye, aunque sea obligatorio por la política de firma |
| | NO | |
| IndividualDataObjectsTimeStamp | SI | Nunca se incluye, aunque sea obligatorio por la política de firma |
| | NO | |

*Tabla 5: Propiedades firmadas generadas por el módulo de firma*

Las clases de acciones de generación de evidencias pueden ser instanciadas por las distintas entidades indicando en cualquier caso:

- URL del prestador de certificación que publica las SP/extSP y resto de servicios de certificación.
- Ruta y contraseña del almacén PKCS#12 con el certificado X509 que se usará para generar la firma electrónica.
- Rol que se indica en la generación de la firma electrónica.
- Identificación del entorno en el que se genera la firma.

Adicionalmente, y dependiendo de la acción a ejecutar en cada caso, es necesario indicar la siguiente información adicional en el momento de creación de las distintas acciones:

- *GeneratePNRO1Action:* Es necesario indicar el mensaje con la información a intercambiar, y el contenido de la extSP que se utilizará en la generación de evidencias durante la ejecución del protocolo.

- *GeneratePNRR1Action, GeneratePNRRO1Action, GeneratePNRRO2Action, GenerateNREAction:* Requieren que se les informe de la evidencia obtenida ya validada (por ejemplo, PNRO1 en caso de *GeneratePNRR1Action*).

Al ejecutarse estas acciones mediante la invocación del método *run()*, inicialmente se obtienen las reglas de generación usando el nivel de compromiso que corresponda y la SP indicada para dicha evidencia en la extSP que se utilice en la ejecución del protocolo. Una vez obtenidas las reglas, se genera la evidencia haciendo uso de implementación realizada en *ofepspplus_signature* para la generación de firmas XAdES.

### Acciones para generación de evidencias de subprotocolos

En la ejecución del subprotocolo de recuperación, el TTP debe generar la evidencia NRE-TTP, para ello se usará la acción implementada en *GenerateNREAción*, que instancia el TTP con información de su rol y el nivel de compromiso correspondiente que permita obtener las reglas de generación de la evidencia.

Las extSP generadas y que se utilizan en la ejecución contienen dos árboles de solución. Uno de ellos corresponde al protocolo principal y el otro incluye la relación de firmas del subprocolo de recuperación. Para el caso del subprotocolo de interrupción no se tendrán en cuenta estos árboles de solución, pero se implementan las acciones necesarias para generación de las evidencias necesarias para la ejecución de dicho subprotocolo.

Para permitir la generación de la firma de la petición de interrupción por parte del Origen se implementa la acción *GenerateAbortAction.* Esta acción genera una firma XAdES, usando el core de firma implementad., Pero a diferencia de las evidencias generadas en el protocolo principal, no se tienen en cuenta reglas de SP. Se trata de una firma XML (XAdES), donde la información a firmar es un nodo XML compuesto del identificador de la ejecución y la acción a realizar (ABORT), y que incluye como propiedades firmadas entre otras, el rol y la fecha de generación de la firma, estas propiedades permiten obtener cuando se solicito la interrupción del protocolo de una determinada ejecución.

La generación de la evidencia de interrupción por parte del TTP corresponde a una *counterSignature* de la firma de petición de interrupción obtenida del Origen. La generación de esta evidencia se implementa en la acción *GenerateAbortTTP* que genera una *counterSignature* de la firma de la petición del Origen, sin tener en cuenta reglas de las SP, pero se incluye como propiedades firmadas

entre otras el rol y la fecha de generación, para permitir obtener cuando se confirmo la interrupción del protocolo de una determinada ejecución.

### *3.4.2.4Validación de Evidencias*

Ante la necesidad de realizar la validación de una evidencia obtenida, las entidades deberán instanciar una de las acciones de validación con los parámetros que correspondan, y ejecutar la acción instanciada para obtener el resultado de la validación.

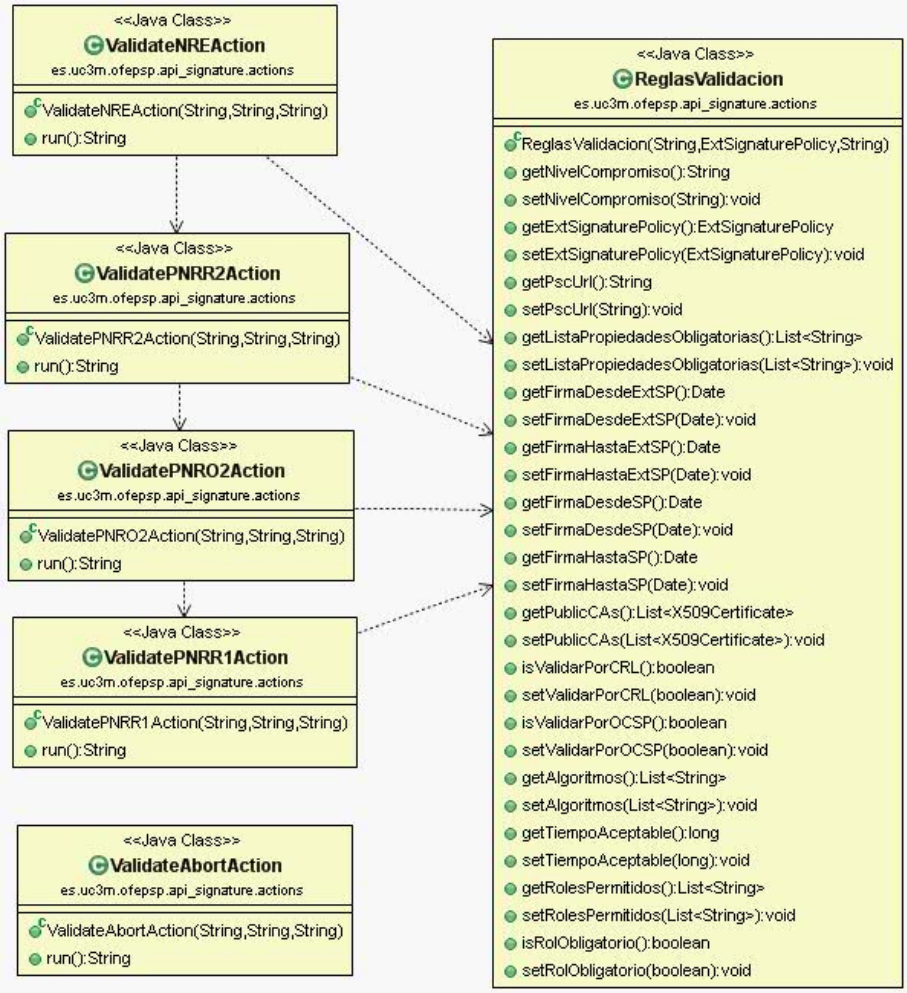



*Figura 17: Diagrama de clases que implementan las acciones de validación de evidencias*

### Acciones para validación de evidencias de protocolo principal

Las acciones implementadas para la validación de evidencias usan la acción implementada en *GetReglasValidacionAction* para obtener las reglas de validación a usar en cada caso. Se permite instanciar esta acción indicando el nivel de compromiso, el contenido de extSP, y la URL del PSC que se usará para obtener la SP necesaria en cada cada caso. Al ejecutar esta acción se obtiene una instancia de la clase de *ReglasValidacion* con las reglas a usar en la validación de la firma. La clase *ReglasValidacion* encapsula tanto la información de las reglas existentes en la SP, así como las condiciones de validación de las firmas dentro de la transacción obtenidas de extSP.

Las clases de acciones de generación de evidencias, *ValidatePNRO1Action, ValidatePNRR1Action, ValidatePNRO2Action, ValidatePNRR2Action, ValidateNREAction,* pueden ser instanciadas por las distintas entidades indicando en cualquier caso:

- Contenido de la evidencia a validar.
- Identificador de la ejecución del protocolo en la que se ha generado la evidencia.
- URL del PSC que se usará para acceder a los servicios de publicación de SP y servicios de validación OCSPResponder y CRL.

Una vez instanciadas, al ejecutarse estas acciones mediante la invocación del método *run()*, se realiza el proceso de validación de la evidencia. La implementación de la validación de evidencias realiza las siguientes comprobaciones para determinar si la evidencia obtenida es correcta:

1. Comprobar que la evidencia corresponde con el identificador de ejecución correcto. Esta comprobación es posible ser realizada ya que el identificación de ejecución forma parte de la información firmada en PNRO1 y PNRR1.
2. Comprobar que la jerarquía de firmas existente cumple con los requisitos del árbol de solución indicado en extSP.
3. Realizar la validación de la evidencia anterior, en caso de no tratarse de la evidencia inicial (PNRO1), se instancia y ejecuta la acción para validar la evidencia anterior. Esto implica que por ejemplo en el proceso de validación de NRE se realice la validación de PNRR2, y a su vez de PNRO2, PNRR1 y PNRO1.
4. Se realiza una validación de todas los valores de firmas (*SignatureValue*) que se encuentran en la evidencia. Esta validación implica comprobar que todas las referencias de la firma son correctas y los valores de los elementos *SignatureValue* son los correctos, aplicando para ello los algoritmos de transformación, algoritmos de hash, y cifrado indicados en las estructuras de firma. Para realizar estas comprobaciones se usa la clase implementada *ValidatorSignatureValue* que utiliza la propia validación de la API de firmas electrónicas implementada en la JDK [JSR105].
5. Se obtienen las reglas de validación y requisitos de extSP y la SP correspondiente. Para ello inicialmente se obtiene el identificador de extSP y nivel de compromiso de las propiedades firmadas, con esta información se obtiene el contenido de extSP realizando la consulta al PSC. Con el contenido de extSP y la identificación del nivel de compromiso se obtienen las

reglas de validación. Este proceso de obtención de reglas implica obtener del PSC el contenido de la SP que aplique en cada caso.

6. Se comprueba que la firma contiene las propiedades firmadas indicadas como obligatorias en las reglas obtenidas.
7. En los casos que las reglas lo indiquen, se comprueba si la firma incluye información sobre el rol, y que dicho rol está incluido en la SP entre los permitidos para el nivel de compromiso.
8. Comprobar que se ha utilizado un algoritmo de firma permitido por las reglas obtenidas.
9. Comprobar que la firma se realizó dentro del periodo de validez de la SP y extSP, utilizando para ello el contenido de la propiedad firmada *SigningTime*.
10. Se obtiene el certificado firmante (X509) y se comprueba que no estaba caducado en el momento de la firma, y que está emitido por una CA reconocida por la SP.
11. Se comprueba el estado de revocación del certificado firmante usando el servicio OCSPResponder y/o consultando la CRL publicada por el PSC. En este punto, una evidencia generada con un certificado determinado puede darse por incorrecta o correcta dependiendo si en las reglas obtenidas se indica que la validación ha realizarse vía CRL o OCSP respectivamente, ya que se han emitido certificados de prueba que no se encuentran en la CRL pero que están revocados.
12. En el caso de no tratarse de PNRO1, se comprueba que el tiempo de creación de la firma se encuentra dentro del tiempo aceptable según el árbol de solución de extSP. Para ello se comparan los valores de las propiedades *SigningTime* de las distintas firmas generadas en cada paso.

Una vez finalizado el proceso de validación, la entidad que ejecuta la acción obtendrá una cadena de texto con el resultado de validación, que tendrá el valor "[OK]" si la validación fue satisfactoria, y en casos de no realizarse una validación correcta el valor "[KO]" seguido de la descripción del error de validación.

### Acciones para validación de evidencias de subprotocolos

En la ejecución del subprotocolo de recuperación, el TTP usará las acciones de validación del protocolo principal descritas en el apartado anterior para la validación de PNRO1, PNRR1, PNRO2, PNRR2.

Para permitir a TTP validar la petición de interrupción generada por el Origen se implementa la acción *ValidateAbortAction.* Esta acción realiza una validación completa de la firma de la petición, incluida la validación del estado de revocación del certificado firmante mediante consulta a la CRL del PSC, pero sin tener en cuenta reglas de SP/extSP al no definirse un árbol de solución para este subprotocolo en las extSP publicadas.

### 3.4.2.5Utilidades comunes implementadas en ofepspplus_signature

A parte de la implementación de los procesos de firma y validación, así como de las clases necesarias para interpretar SP y extSP; dentro del módulo *ofepsplus_signature* se encuentran implementadas las clases de apoyo y utilidades comunes a todas las entidades. Entre esta funcionalidad se encuentra implementado:

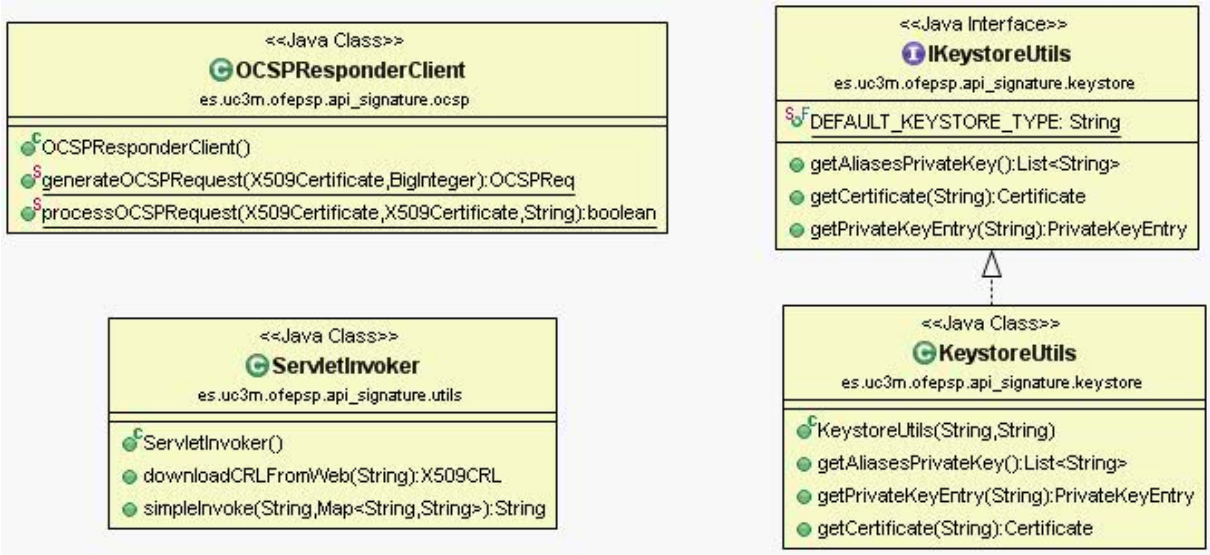



*Figura 18: Diagrama de clases de las utilidades comunes implementadas en ofepspplus_signature*

- ***OCSPResponderClient***. Implementa el cliente OCSP que permite realizar peticiones a un OCSPResponder y obtener la respuesta según [RFC2560].

- ***ServletInvoker***. Clase de utilidad para realizar llamadas a Servlets desplegados por otras entidades. Será utilizado por las entidades involucradas para consultar las SP y extSP usando los servicios publicados por el PSC. Adicionalmente también proporciona la implementación de la descarga de CRLs desde una ruta determinada.

- ***KeystoreUtils***. Permite listar los alias existentes en un PKCS#12, así como obtener la parte pública (*X509Certificate)* y las claves privadas existentes en el almacén de claves a partir de sus alias. Para acceder al contenido de un PKCS#12, será necesario instanciar esta clase de utilidad indicando tanto la ruta local del fichero, como la contraseña del mismo.

El resto de clases y paquetes existentes en este componente, implementan las constantes con los valores comunes, el dialecto SQLite para acceso a datos, y proporciona las clases de utilidades comunes para realizar codificaciones y decodificaciones en Base 64, ejecutar algoritmos de Hash, operaciones con estructuras XML, etc.

## 3.5 *ofepspplus_psc*, Prestador de Servicios de Certificación

Se realiza la implementación de una aplicación Web, que cubra la funcionalidad de un PSC. Este prestador publicará los servicios de certificación necesarios para la ejecución del protocolo.

### 3.5.1 Análisis funcional

Este módulo cubrirá la funcionalidad necesaria, para las distintas entidades involucradas en el protocolo, de servicios de certificación. Esta funcionalidad se resume en los siguientes servicios de certificación que se deben proporcionar:

- **Emisión y publicación de SP y extSP**: Dispondrá de una interfaz y la implementación necesaria para generar y publicar SP y extSP, que posteriormente puedan ser utilizadas por las entidades del protocolo durante la ejecución del protocolo principal o subprotocolo de recuperación.

- **Emisión de certificados reconocidos:** Este PSC será el responsable de la emisión de los certificados que se indicarán como reconocidos en las SP. Para ello emitirá los certificados que requiera cada entidad del protocolo.

- **Publicación de servicios de validación:** El PSC debe proporcionar a las distintas entidades los servicios de validación necesarios para comprobar el estado de revocación de los certificados emitidos. Para ello publicará las CRLs necesarias, a la vez que proporcionará un servicio de OCSPResponder.

### 3.5.2 Implementación del PSC

En los siguientes apartados se describe como se ha abordado la implementación, partiendo de que el PSC se implementa como una aplicación Web que presta los servicios del certificación. Dentro de proyecto Web del PSC se han implementado las clases incluidas en el siguiente diagrama.

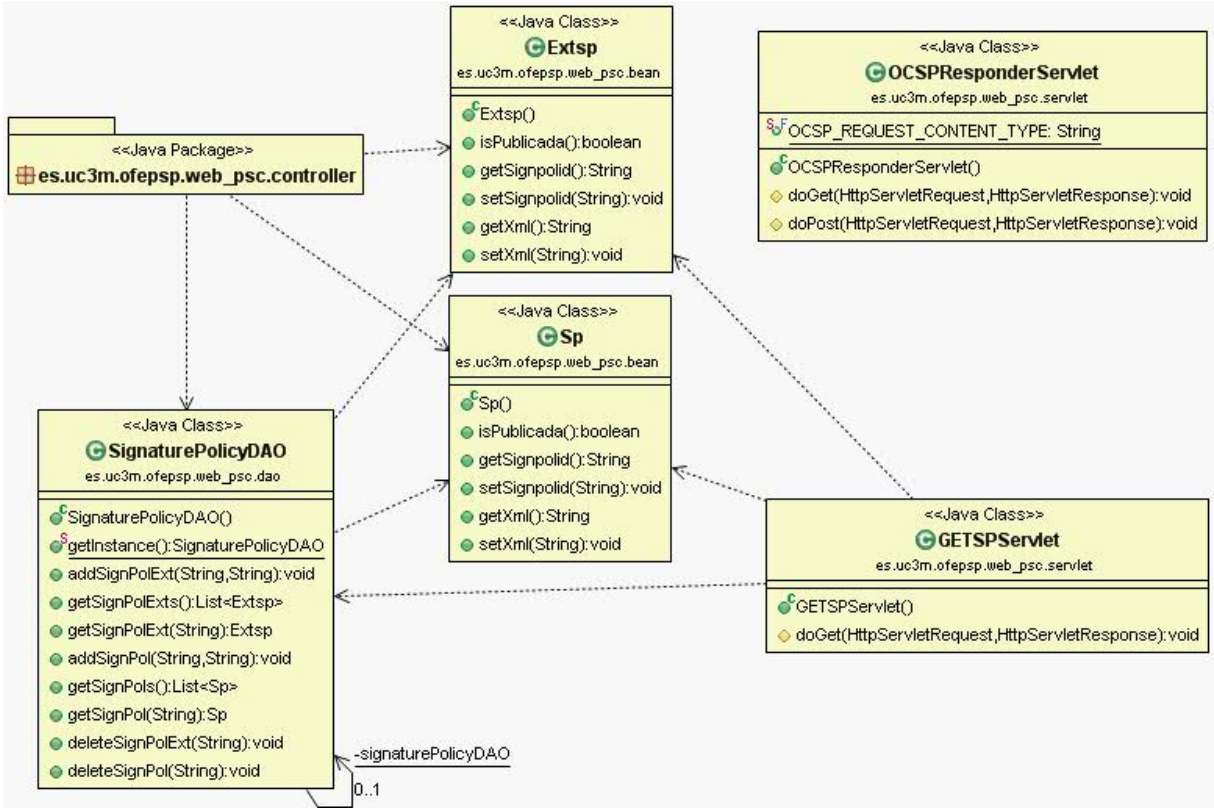



*Figura 19: Diagrama de clases del módulo Web que publica los servicios del PSC*

La finalidad de estas clases se describe en los siguientes apartados, en los que se detalla la implementación del módulo Web que publica los servicios del PSC.

### *3.5.2.1 Modelo de datos y persistencia*

La aplicación WEB dispone de un base de datos tipo [SQLite] con las tablas necesarias para la gestión de las SP y extSP por parte del PSC. En esta base de datos existen dos tablas: SP y EXTSP.

| **TABLA SP**<br>Tabla en la que se almacenan las SP emitidas por el PSC | | |
|---|---|---|
| **CAMPO** | **TIPO** | **DESCRIPCION** |
| SIGNPOLID | CHAR(255) | Identificador de la política de firma |
| XML | TEXT | Implementación XML de la política de firma acorde a [TR102038] |

| **TABLA EXTSP**<br>Tabla en la que se almacenan las extSP emitidas por el PSC | | |
|---|---|---|
| **CAMPO** | **TIPO** | **DESCRIPCION** |
| EXTSIGNPOLID | CHAR(255) | Identificador de la política de firma extendida |
| XML | TEXT | Implementación XML de la política de firma extendida acorde al XSD definido en el apéndice E de [JLHA] |

El módulo que implementa la funcionalidad del PSC, usa el ORM *Hibernate* para realizar las operaciones de alta, baja y peticiones de consulta necesarias sobre esta tablas. Para ello se implementan dos clases que permiten representar en objetos planos Java (POJOs) la información de las tablas:

```
es.uc3m.ofepsp.web_psc.bean.SP ↔ Tabla SP
es.uc3m.ofepsp.web_psc.bean.SPEXT ↔ Tabla EXPSP
```

Para la realización de operaciones sobre la base de datos, se implementa un DAO en la clase S*ignaturePolicyDAO* con las operaciones necesarias para obtener, eliminar y dar de alta tanto SP como extSP.

### *3.5.2.2 Gestión de certificados reconocidos*

Se utiliza la herramienta [XCA] para generar un certificado autofirmado que representa la CA, por lo que tendrá como uso de clave al menos los valores necesarios para no repudio, firma de certificados, y firma de CRL.

Dicha CA emitirá los distintos certificados requeridos por las distintas entidades, que usarán para la generación de las evidencias en la ejecución del protocolo. En un principio, y para poder realizar ejecuciones del protocolo en las que se valide de forma satisfactoria, o no, el estado de revocación de los certificados, se emiten los siguientes certificados:

- **Certificados del PSC**

| Emitido a [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES**<br>Emitido por [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES** | |
|---|---|
| Certificado auto-firmado (CA) utilizado para la emisión del resto de certificados y firma de CRL. | |
| **Número de serie** | HEX: 01 |
| **Usos de clave** | `Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Certificate Sign, CRL Sign` |
| **Estado de validez** | No caducado / No revocado |

| Emitido a [RFC2253]: **CN=OFEPSP_TSP,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES**<br>Emitido por [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES** | |
|---|---|
| Certificado utilizado por el PSC para prestar servicios como la emisión de políticas de firma y firma de componentes software. | |
| **Número de serie** | HEX: 02 |
| **Usos de clave** | `Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Code Signing, Time Stamping, OCSP Signing` |
| **Estado de validez** | No caducado / No revocado |

- **Certificados emitidos al Receptor**

| Emitido a [RFC2253]: **CN=OFEPSP_RECEIVER_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES**<br>Emitido por [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES** | |
|---|---|
| Certificado a usar por el Receptor para la generación de evidencias en la ejecución del protocolo, también puede ser usado para la firma de componentes software como por ejemplo el Applet a publicar, y la firma de peticiones OCSP. | |
| **Número de serie** | HEX:13 |
| **Usos de clave** | `Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Code Signing, OCSP Signing` |
| **Estado de validez** | No caducado / No revocado |

| Emitido a [RFC2253]: **CN=OFEPSP_RECEIVER_2,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES**<br>Emitido por [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES** | |
|---|---|
| Certificado a usar por el Receptor para la generación de evidencias en la ejecución del protocolo, también puede ser usado para la firma de componentes software como por ejemplo el Applet a publicar, y la firma de peticiones OCSP. | |
| **Número de serie** | HEX:14 |
| **Usos de clave** | `Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Code Signing, OCSP Signing` |
| **Estado de validez** | No caducado / **Revocado antes de la emisión de CRL** |

| Emitido a [RFC2253]: **CN=OFEPSP_RECEIVER_3,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES**<br>Emitido por [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES** |
|---|

| Certificado a usar por el Receptor para la generación de evidencias en la ejecución del protocolo, también puede ser usado para la firma de componentes software como por ejemplo el Applet a publicar, y la firma de peticiones OCSP. | |
|---|---|
| **Número de serie** | HEX: 12 |
| **Usos de clave** | `Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Code Signing, OCSP Signing` |
| **Estado de validez** | No caducado / **Revocado posteriormente a la emisión de CRL** |

- **Certificados emitidos al Origen**

| Emitido a [RFC2253]: **CN=OFEPSP_ORIGIN_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES**<br>Emitido por [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES** | |
|---|---|
| Certificado a usar por el Origen para la generación de evidencias en la ejecución del protocolo, también usado para la firma de peticiones OCSP. | |
| **Número de serie** | HEX: 0D |
| **Usos de clave** | `Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Certificate Sign, OCSP Signing` |
| **Estado de validez** | No caducado / No revocado |

| Emitido a [RFC2253]: **CN=OFEPSP_ORIGIN_2,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES**<br>Emitido por [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES** | |
|---|---|
| Certificado a usar por el Origen para la generación de evidencias en la ejecución del protocolo, también usado para la firma de peticiones OCSP. | |
| **Número de serie** | HEX: 0E |
| **Usos de clave** | `Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Certificate Sign, OCSP Signing` |
| **Estado de validez** | No caducado / **Revocado antes de la emisión de CRL** |

| Emitido a [RFC2253]: **CN=OFEPSP_ORIGIN_3,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES**<br>Emitido por [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES** | |
|---|---|
| Certificado a usar por el Origen para la generación de evidencias en la ejecución del protocolo, también usado para la firma de peticiones OCSP. | |
| **Número de serie** | HEX: 0F |
| **Usos de clave** | `Digital Signature, Non Repudiation, Key` |

| | |
|---|---|
| | Encipherment, Data Encipherment, Key Agreement, Certificate Sign, OCSP Signing |
| **Estado de validez** | No caducado / **Revocado posteriormente a la emisión de CRL** |

- **Certificados emitidos a TTP**

| Emitido a [RFC2253]: **CN=OFEPSP_TTP_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES**<br>Emitido por [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES** | |
|---|---|
| Certificado a usar por la tercera entidad de confianzada para la generación de evidencias en la ejecución del protocolo, también usado para la firma de peticiones OCSP. | |
| **Número de serie** | HEX: 0B |
| **Usos de clave** | Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Certificate Sign, OCSP Signing, Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Code Signing, OCSP Signing |
| **Estado de validez** | No caducado / **Revocado antes de la emisión de CRL** |

| Emitido a [RFC2253]: **CN=OFEPSP_TTP_2,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES**<br>Emitido por [RFC2253]: **CN=OFEPSP_CA_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES** | |
|---|---|
| Certificado a usar por la tercera entidad de confianzada para la generación de evidencias en la ejecución del protocolo, también usado para la firma de peticiones OCSP. También será usado para la firma de los componentes software (Applet) que distribuye el Receptor. | |
| **Número de serie** | HEX: 0C |
| **Usos de clave** | Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Certificate Sign, OCSP Signing, Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement, Code Signing, OCSP Signing |
| **Estado de validez** | No caducado / No Revocado |

Con los certificados emitidos se permite la ejecución del protocolo en distintas circunstancias, pero podrían emitirse otros certificados, incluso de otras CA/subCAs y usar dichos certificados en la

generación de evidencias durante la ejecución del protocolo. Para ello hay que tener en cuenta que para que las evidencias generadas con un cierto certificado sean válidas, dicho certificado ha de ser emitido por una entidad reconocida por SP y deben estar disponibles los servicios de validación (CRL/OCSP) para poder consultar su estado de revocación.

### *3.5.2.3Emisión y publicación de CRL*

Se usará la herramienta XCA para generar la CRL con las siguientes entradas:

| **Número de serie** | **[RFC2253]** |
|---|---|
| HEX: 0B | CN=OFEPSP_TTP_1,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES |
| HEX: 14 | CN=OFEPSP_RECEIVER_2,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES |
| HEX: 0E | CN=OFEPSP_ORIGIN_2,OU=UC3M,O=UC3M,L=Leganes,ST=Madrid,C=ES |

*Tabla 6: Certificados incluidos en CRL de pruebas*

Estos números de serie corresponden a los certificados que se encuentran revocados en el momento de emisión/publicación de la CRL. Dicha lista de revocación se encuentra disponible para todas las entidades involucradas en la ejecución del protocolo y pueden obtenerla para la consulta del estado de revocación de certificados en la siguiente URL:

```
http://XX.XX.XX.XX:PUERTO/web-psc/public/OFEPSP_CA.crl
```

Donde XX.XX.XX.XX es la dirección IP del servidor de aplicaciones Web en el que se ha desplegado la aplicación con los servicios del PSC, por el PUERTO indicado. Las distintas entidades, podrán consultar la CRL en la ruta indicada para conocer el estado de revocación de los certificados emitidos por el PSC.

### *3.5.2.4Servicio OCSPResponder*

La aplicación Web del PSC, publica un OSCPResponder implementado en la clase *OCSPResponderServlet* según [RFC2560], y se desplegará con la aplicación Web del PSC, al configurarse en el descriptor *web.xml* del módulo Web dicho *Servlet*:

```
<servlet>
      <servlet-name>ServidorOCSP</servlet-name>
      <servlet-class>
          es.uc3m.seti.ofepsp.tsp.servlet.OCSPResponderServlet
      </servlet-class>
      <load-on-startup>3</load-on-startup>
```

```
    </servlet>
    <servlet-mapping>
        <servlet-name>ServidorOCSP</servlet-name>
        <url-pattern>/OCSPResponder</url-pattern>
    </servlet-mapping>
```

De esta forma el OCSPResponder se encuentra disponible para la obtención del estado de revocación de los certificados por las distintas entidades involucradas en la ejecución del protocolo, aceptándose peticiones de consulta en la siguiente ruta:

```
http://XX.XX.XX.XX:PUERTO/web-psc/OCSPResponder
```

Donde *XX.XX.XX.XX* es la dirección IP del servidor de aplicaciones Web en el que se ha desplegado la aplicación Web del PSC, por el *PUERTO* en el que se despliegue la aplicación Web.

Para determinar el estado de revocación de los certificados, el servicio OCSP inicialmente consulta la CRL publicada en el mismo servidor. En los casos que el número de serie consultado se encuentre en dicha CRL el OCSP devolverá el estado revocado para la fecha de revocación indicada en la CRL. En los casos que el número de serie del certificado consultado no se encuentre en la CRL, el OCSP devolverá un estado revocado si se trata de uno de los números de serie revocados posteriormente según el estado indicado en los certificados emitidos por el PSC (los certificados de prueba emitidos por el PSC se numeran en el apartado 3.5.2.2 de la presente memoria).

### *3.5.2.5Gestión y generación de Políticas de Firma*

La implementación de esta funcionalidad permite generar y publicar SP que se referenciarán desde las extSP y dispondrán de los niveles de compromiso (*CommitmentRules)* necesarios para la simulación del protocolo, como por ejemplo la política de firma en XML incluida en el *ANEXO A.1*.

Para la gestión y generación de nuevas SP se implementa la lógica de negocio necesaria en el paquete *es.uc3m.ofepsp.web_psc.controller y* se crea una interfaz de usuario compuesta por las siguientes vistas existentes en el módulo Web:

***gestion_sp.xhtml***: Muestra un listado con todas las SP existentes, permitiendo las acciones de descarga de implementación XML de cada una de ellas, y la posibilidad de eliminar las SP.


*Figura 20: Interfaz gráfica para la gestión de SP*

En la clase controlador *SPListarController* se implementa la obtención del listado en su método *getSps()*. Del mismo modo, en este controlador se implementan las acciones de descarga y borrado de políticas en los métodos *getSpDownload* y *onSpEliminar* respectivamente.

*alta_sp.xhtml:*Desde esta vista se permite introducir la información para generar una nueva SP, y desde la que se puede ejecutar la acción de creación y publicación de la nueva SP. Se compone de un formulario en el que se puede indicar la siguiente información:


*Figura 21: Información de identificación de SP en alta de nueva SP*

Información de la identificación de la Política, permite indicar el identificador único, fecha de emisión y campo de aplicación.


*Figura 22: Información de política de validación en alta de SP*

Información básica de la política de validación, pudiendo indicar la fecha inicial y final del periodo en el que puede usarse la SP.

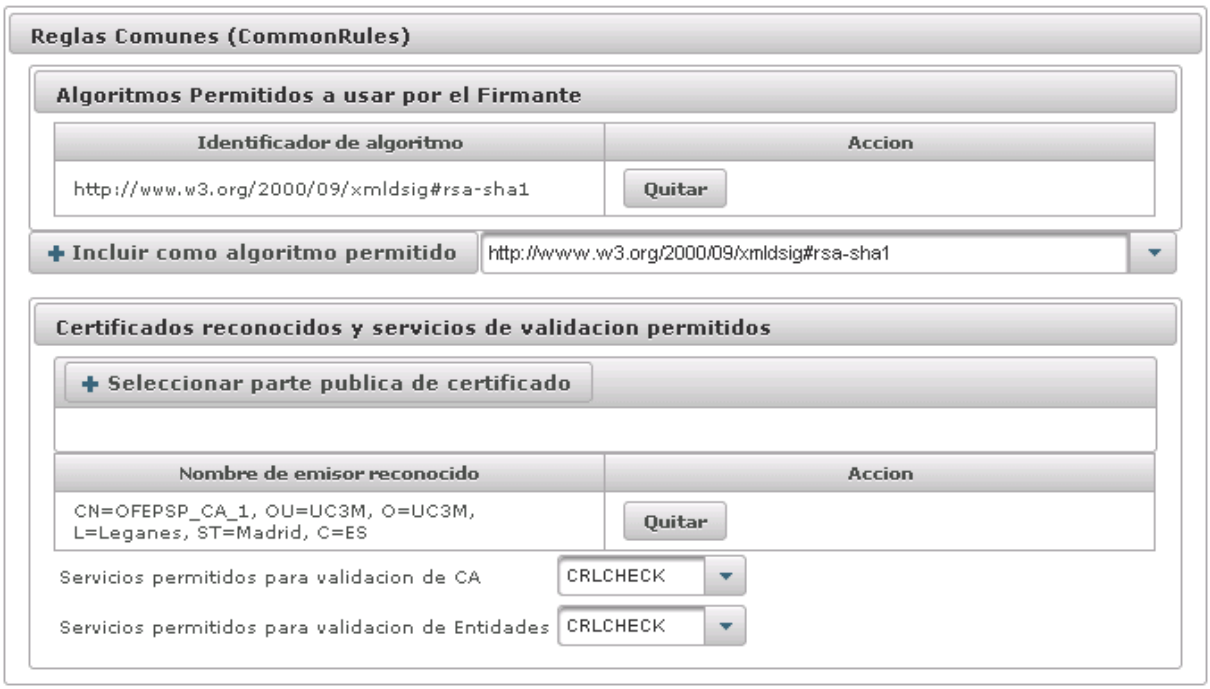



*Figura 23: Información de reglas comunes en alta de SP*

Reglas comunes, se permiten indicar los algoritmos de firma permitidos. Y los tipos de certificado permitidos, pudiéndose indicar emisores de certificados (CA) reconocidos a partir de estructuras X509, así como indicar los métodos de validación de estados de revocación (CRL y/o OCSP).


*Figura 24: Selección de nivel de compromiso en alta de SP*

En este mismo formulario se incluyen las opciones para indicar las reglas de cada uno de los niveles de compromiso que aplican en el protocolo.

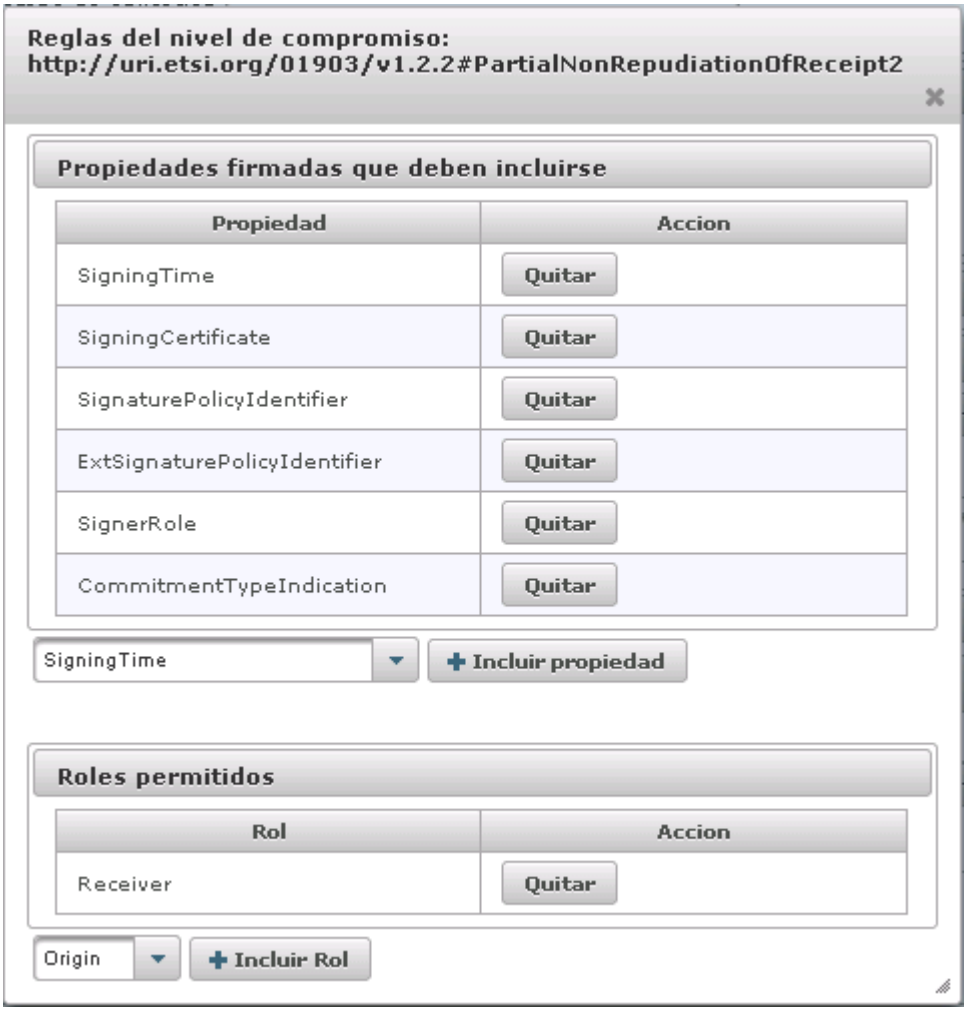



*Figura 25: Información de reglas por nivel de compromiso en alta de SP*

Por cada nivel de compromiso se permite indicar en la creación de la SP las reglas relativas a propiedades firmadas obligatorias y roles permitidos.

La vista *alta_sp.xhtml*, permite ejecutar la acción de alta y publicación de la SP con la información introducida en su formulario. La implementación de la lógica de negocio utilizada en el alta se encuentra implementada en la clase controlador *SPAltaController*. En este controlador se implementan los métodos necesarios para obtener los valores permitidos que se pueden indicar en la creación de la nueva SP, como por ejemplo, servicios de validación y algoritmos de firma, así como métodos para la obtención de la información introducida en el formulario. El método *onPublicarPolitica()* existente en *SPAltaController* será el encargado de crear la nueva política, este método realiza el alta y publicación ejecutando los siguientes pasos:

1. Validar los datos de entrada, comprobando que los campos obligatorios se han introducido y que son válidos. Si se ecnuentra algún error el controlador devuelve la descripción de dicho error a la vista.
2. Utilizando las clases del módulo *ofepspplus_signature*, genera la SP en formato XML.
3. Con la información del certificado del PSC, se incluye el nombre del emisor en la nueva SP.
4. Una vez completado el XML se calcula el Hash de dicho XML para indicarlo en la SP.
5. Se añade la nueva SP en la tabla SP de la Base de Datos del PSC.

#### *3.5.2.6Gestión y generación de Políticas de Firma Extendidas*

Se implementa dentro del módulo Web un frontal que permite crear nuevas extSP en formato XML para su uso en la ejecución del protocolo. Para ello se limitará a la generación de extSP con los siguientes elementos:

- Datos de identificación de la política de firma extendida: identificador, fecha de emisión y nombre del emisor.
- Política de validación:
  - Periodo de firma permitido (fecha inicio y fin)
  - Dos Arboles de solución:
    - Árbol de solución con el conjunto de firmas a generar en la ejecución del protocolo principal de OFEPSP+.
    - Árbol de solución con el conjunto de firmas a generar en la ejecución de subprotocolo de recuperación de OFEPSP+.

Para la gestión y generación de nuevas extSP se implementa la lógica de negocio necesaria en el paquete *es.uc3m.ofepsp.web_psc.controller y* se crea una interfaz de usuario compuesta por las siguientes vistas existentes en el módulo Web:

***gestion_extsp.xhtml***: Muestra un listado con todas las extSP existentes, permitiendo las acciones de descarga de implementación XML de cada una de ellas, y la posibilidad de eliminar las extSP existentes.


*Figura 26: Interfaz gráfica para la gestión de extSP*

En la clase controlador *EXTSPListarController* se implementa la obtención del listado en su método *getExtsps()*. Del mismo modo, en este controlador se implementan las acciones de descarga y borrado de políticas en los métodos *getExtspDownload* y *onExtspEliminar* respectivamente.

*alta_extsp.xhtml*:Desde esta vista se permite introducir la información para generar una nueva extSP, y permite ejecutar la acción de creación y publicación de la nueva extSP. Se compone de un formulario en el que se puede indicar la información básica de la extSP (identificador y fecha de creacción), y la información de validación permitiendo indicar el periodo de validez el las propiedades de los nodos de firma de los árboles de solución.

De cada nodo de firma se permite indicar la SP que aplica y el tiempo aceptable de creación respecto al resto de nodos del árbol de solución. El resto de propiedades, así como la estructura de los árboles de solución no se permite modificar, siendo este el mismo y corresponde a los árboles de solución necesarios para la ejecución del protocolo. En el formulario se muestran estos árboles y se permiten seleccionar los nodos para la modificación de sus valores.

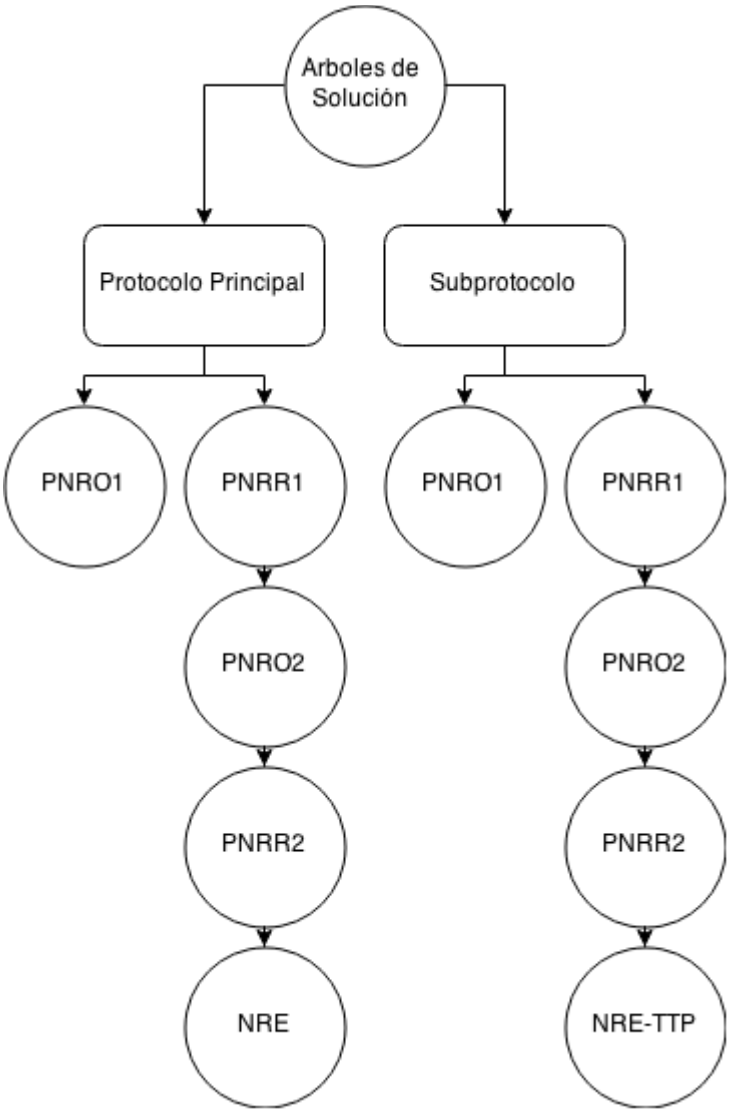



*Figura 27: Árboles de solución de las extSP creadas por el PSC*

Como ya se ha comentado de cada nodo se permite modificar la SP a aplicar y los tiempos de creación aceptables, al indicar esta información de los nodos PNRO1, PNRR1, PNRO2 o PNRR2, los valores introducidos quedarán reflejados para el nodo en ambos arboles. Por otro lado, se permite indicar las propiedades de NRE y NRE-TTP, quedando reflejadas estas propiedades en el nodo del árbol que corresponda.

La vista *alta_extsp.xhtml*, permite ejecutar la acción de alta y publicación de la extSP con la información introducida en su formulario. La implementación de la lógica de negocio utilizada en el alta se encuentra implementada en la clase controlador *SPEXTAltaController.* En este controlador se implementan los métodos necesarios para obtener los valores permitidos que se pueden indicar en la creación de la nueva extSP, como son las SP que se pueden seleccionar, así como métodos para la obtención de la información introducida en el formulario. El método *onPublicarPolitica()* existente en

*SPEXTAltaController* será el encargado de crear la nueva política, este método realiza el alta y publicación ejecutando los siguientes pasos:

1. Validar los datos de entrada, comprobando que los campos obligatorios se han introducido y que son válidos. Si se encuentra algún error el controlador devuelve la descripción de dicho error a la vista.
2. Utilizando las clases del módulo *ofepspplus_signature*, genera la extSP en formato XML.
3. Se añade la nueva extSP en la tabla EXTSP de la Base de Datos del PSC.

#### *3.5.2.7Servicio para la publicación de SP y extSP*

Tanto las SP, como las extSP que se generen desde la aplicación Web del PSC, se encuentran accesibles para el resto de entidades involucradas en la ejecución del protocolo de intercambio. Para ello, la propia aplicación Web del PSC dispone de un servicio Web que se puede ser invocado para obtener las distintas SP y extSP en formato XML. Este servicio se encuentra implementado en el servlet *GetSPServlet* que se publica en la siguiente ruta:

```
http://XX.XX.XX.XX:PUERTO/web-psc/GETSP
```

Donde *XX.XX.XX.XX* es la dirección IP del servidor de aplicaciones Web en el que se encuentra desplegados los servicios del TSP, por el *PUERTO* indicado.

La publicación de este servicio permite consultar las políticas existentes con llamadas al servlet por GET indicado el identificador "urisp" o "uriextsp" para la consulta de SP y extSP respectivamente. Como funcionalidad necesaria, si el usuario del servicio indica como identificador el valor "LIST", el servicio devuelve la lista de identificadores (URIs) de Políticas de Firma o Políticas de Firma disponibles.

A continuación se muestran ejemplos de los distintos casos de uso del servicio publicado por el PSC para permitir consulta de políticas:

**Ejemplo de listado de Políticas de Firma disponibles:**

*Petición:*

```
http://XX.XX.XX.XX:PUERTO/web-psc/GETSP?urisp=LIST
```

*Respuesta:*

```
<spIds>
      <spId>politica1</spId>
      <spId>politica2</spId>
      <spId>politica3</spId>
      [...]
</spIds>
```

**Ejemplo de listado de Políticas de Firma Extendidas disponibles:**

*Petición:*

```
http://XX.XX.XX.XX:PUERTO/web-psc/GETSP?uriextsp=LIST
```

*Respuesta:*

```
<extSpIds>
      <extSpId>ext1</extSpId>
      <extSpId>ext2</extSpId>
      <extSpId>ext3</extSpId>
      [...]
</extSpIds>
```

**Ejemplo de obtención de una Política de Firma:**

*Petición:*

```
http://XX.XX.XX.XX:PUERTO/web-psc/GETSP?urisp=politica1
```

*Respuesta:*

```
<ns3:signaturePolicyType       xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"
      xmlns="http://uri.etsi.org/2038/v1.1.1#"
      xmlns:ns3="http://uri.etsi.org/01903/v1.3.2#">
      [...]
      <SignPolicyInfo>
            <SignPolicyIdentifier>
                  <ns3:Identifier>politica1</ns3:Identifier>
            </SignPolicyIdentifier>
            [...]
      </SignPolicyInfo>
      <ns2:Signature>
      [...]
      </ns2:Signature>
</ns3:signaturePolicyType>
```

**Ejemplo de obtención de una Política de Firma Extendida:**

*Petición:*

```
http://XX.XX.XX.XX:PUERTO/web-psc/GETSP?uriextsp=ext1
```

*Respuesta:*

```
<ns2:extSignaturePolicy
      xmlns:ns2="http://jlopez.thesis.uc3m.es/ETS-
      ExtendedElectronicSignaturePolicies-97Syntax">
      <ns2:extSignPolicyInfo>
            <ns2:extSignPolicyIdentifier>
                  <ns2:extSignPolicyId>ext1</ns2:extSignPolicyId>
                  <ns2:dateOfIssue>[...]</ns2:dateOfIssue>
            </ns2:extSignPolicyIdentifier>
            <ns2:extSignValidationPolicy>
                  [...]
```

```
            </ns2:extSignValidationPolicy>
      </ns2:extSignPolicyInfo>
</ns2:extSignaturePolicy>
```

# 3.6 *ofepspplus_buyerApplet*, Componente software del Origen

## 3.6.1 Análisis funcional

Este componente debe cubrir la funcionalidad del Origen, al realizarse la implementación del protocolo siguiendo un modelo B2C, este módulo debe ser una aplicación cliente que ejecute en los distintos entornos sel Origen y que cubra sus requisitos funcionales:

- Acceder a los servicios del PSC, tanto para la consulta de SP y extSP, como para acceder a los servicios de validación (OCSP y CRL)
- Generar firmas electrónicas basadas en las extSP y las reglas de las SP.
- Generar peticiones de interrupción firmadas.
- Validar firmas electrónicas comprobando que cumplen con los requisitos dentro de la transacción establecidos en la extSP, así como comprobando que cumple con las reglas de la SP.
- Interactuar con el resto de entidades para poder realizar el intercambio de evidencias.
- Ejecutarse desde dos entornos distintos (OE1, OE2).

## 3.6.2 Implementación

Para proporcionar la funcionalidad necesaria del Origen, se implementa un Applet Java que se ejecuta en el navegador de la entidad Origen. Este Applet lo proporciona el Receptor y TTP, por lo que será desplegado por parte de los módulos Web implementados por *ofepspplus_vendor* y *ofepspplus_ttp*..

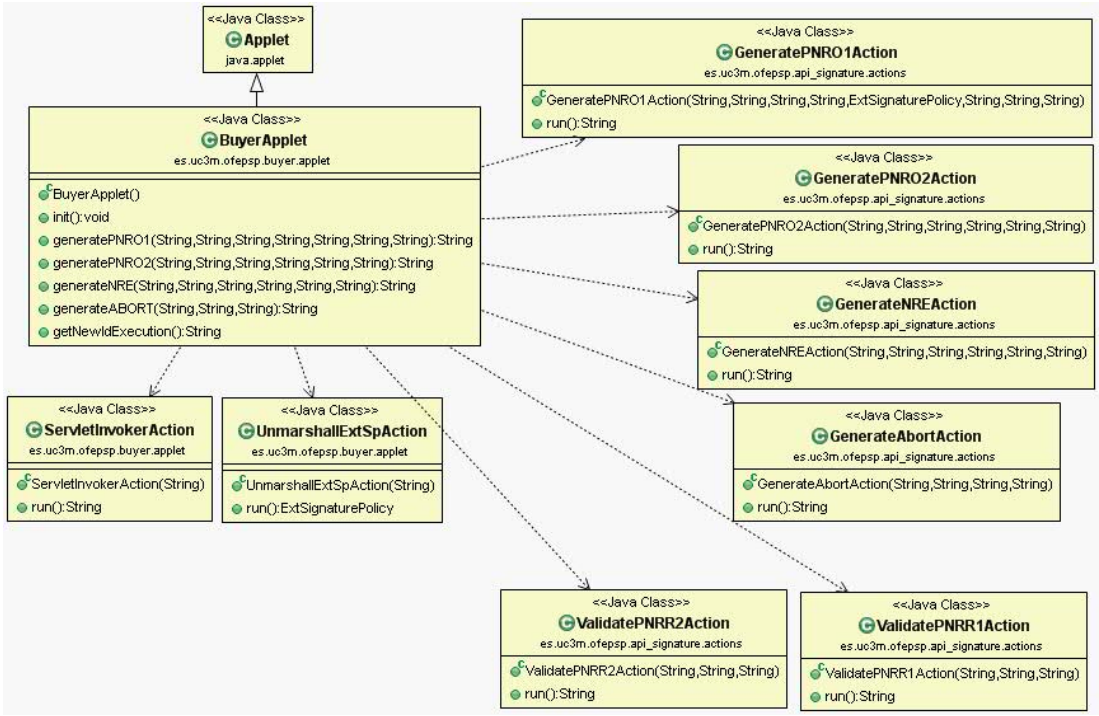



*Figura 28: Diagrama de clases del componente cliente*

### *3.6.2.1Validación y generación de evidencias en Origen*

La clase *BuyerApplet i*mplementa el componente sin interfaz gráfica que se ejecuta dentro del contexto de un navegador Web. Dispone de los métodos necesarios para la validación y generación de evidencias desde un navegador Web:

- **Método getNewIdExecution**: Permite al Origen generar un identificador único para una nueva ejecución del protocolo antes de generar PNRO1.

- **Método generatePNRO1**: Permite al Origen iniciar la ejecución del protocolo. A partir de extspURI, se obtiene el contenido de la política de firma extendida del prestador publicado en rutaPSC mediante el uso de las clases ServletInvokerAction y UnmarshallExtSpAction. Una vez obtenido del contenido de la política de firma extendida, se genera PNRO1 con la clave privada indicada usando las acciones implementadas en el módulo *ofepspplus_signature*. Tiene como entradas:
  - *extspURI*: Identificador de la política de firma extendida que se utilizará para obtener los requisitos y reglas de las evidencias a generar.
  - *msg*: Mensaje con los datos a intercambiar.
  - *idExecution*: Identificador de la ejecución del protocolo, que anteriormente ha asignado el Origen.
  - *pkcs12path*: Ruta al fichero con el PKCS#12 que contiene la clave privada con la que el Origen generará la envidencia PNRO1.

  - *pkcs12pass*: Contraseña para acceder al PKCS#12.
  - *rutaPSC*: Ruta en la que se encuentra desplegado el prestador de servicios que se usará para obtener las políticas de firma y acceder a los servicios de validación.
  - *entorno*: Identificación del entorno en el que se está ejecutando el Applet.

- **Método generatePNRO2**: Este método inicialmente realiza la validación de PNRR1 usando las acciones implementadas en el módulo *ofepspplus_signature*. Si PNRR1 es validado correctamente se genera PNRO2 con la clave privada indicada usando las acciones implementadas en el módulo *ofepspplus_signature*. Tiene como entradas:
  - *pnrr1*: Evidencia PNRR1 obtenida.
  - *idExecution*: Identificador de la ejecución del protocolo, que anteriormente ha asignado el Origen.
  - *pkcs12path*: Ruta al fichero con el PKCS#12 que contiene la clave privada con la que el Origen generará la envidencia PNRO2.
  - *pkcs12pass*: Contraseña para acceder al PKCS#12.
  - *rutaPSC*: Ruta en la que se encuentra desplegado el prestador de servicios que se usará para obtener las políticas de firma y acceder a los servicios de validación.

- **Método generateNRE**: Este método inicialmente realiza la validación de PNRR2 usando las acciones implementadas en el módulo *ofepspplus_signature*. Si PNRR2 es validado correctamente se genera NRE con la clave privada indicada usando las acciones implementadas en el módulo *ofepspplus_signature*. Tiene como entradas:
  - *pnrr2*: Evidencia PNRR2 obtenida.
  - *idExecution*: Identificador de la ejecución del protocolo, que anteriormente ha asignado el Origen.
  - *pkcs12path*: Ruta al fichero con el PKCS#12 que contiene la clave privada con la que el Origen generará la evidencia NRE.
  - *pkcs12pass*: Contraseña para acceder al PKCS#12.
  - *rutaPSC*: Ruta en la que se encuentra desplegado el prestador de servicios que se usará para obtener las políticas de firma y acceder a los servicios de validación.

- **Método generateABORT**: Permite al Origen iniciar la ejecución del subprotocolo de interrupción generando la firma de la petición de interrupción de una ejecución determinada usando las acciones implementadas en el módulo *ofepspplus_signature*. Tiene como entradas:
  - *idExecution*: Identificador de la ejecución a interrumpir.
  - *pkcs12path*: Ruta al fichero con el PKCS#12 que contiene la clave privada con la que el Origen firmará la petición.
  - *pkcs12pass*: Contraseña para acceder al PKCS#12.

Desde un Applet Java no es posible realizar una llamada a un *servlet* publicado por una entidad distinta a la que publica dicho Applet. Esto es un problema para la ejecución del protocolo, ya que el Origen debe poder realizar llamadas al prestador de servicios desde el Applet publicado por el Receptor. Para solventarlo, se implementa en *ServletInvokerAction* una acción con privilegios que permita realizar este tipo de llamadas. Será utilizado por el Applet para poder obtener el contenido de la política de firma extendida antes de iniciar la ejecución del protocolo. Para ejecutar esta acción, es

necesario crear una instancia de la misma indicando la URL a la que invocar. Al ejecutarse la acción, *run()*, se obtiene la política de firma extendida publicada por el prestador en XML.

Del mismo modo, desde un Applet Java no es posible realizar unmarshall con JAXB de XMLs con referencias a xsds externos. Para solventarlo, se implementa en *UnmarshallExtSpAction* una acción con privilegios que permita realizar este tipo de operaciones. Para ejecutar esta acción se debe crear una instancia de la misma indicando la política de firma extendida en XML Al ejecutarse la acción, mediantela invocación del método *run()*, se obtiene una instancia de `es.uc3m.ofepsp.api_signature.jaxb.extsignaturepolicy.ExtSignaturePolicy` con la información de la extSP.

#### *3.6.2.2Ejecución en distintos entornos (OE1, OE2)*

Para permitir que el Origen pueda interactuar con el resto de entidades en dos entornos distintos durante la ejecución del protocolo principal (OE1, OE2), el Applet será capaz de obtener información sobre el entorno local en el que se encuentra ejecutado, de esta forma se identifica cada entorno por la IP física de la interfaz de red que utiliza el Origen para comunicarse con el resto de entidades concatenado con la identificación del navegador que este utilizando para la ejecución del cliente.

Adicionalmente, en las reglas a seguir por el firmante en las políticas de firma emitidas por el PSC, se puede obligar para los niveles de compromiso que sean adecuados, a introducir como elemento firmado la propiedad *SignatureProductionPlace* cuyo valor informa del entorno en el que se generó la evidencia.

## 3.7 *ofepspplus_vendor*, aplicación Web del Receptor

### 3.7.1 Análisis funcional

Esta aplicación Web permite al Origen interacturar con el Receptor, y debe realizar las acciones de la entidad Receptor del protocolo. Al implementarse el protocolo siguiendo un modelo B2C, se implementará como aplicación Web que cumpla los siguientes requisitos:

- Permitir al Origen iniciar la ejecución del protocolo, así como permitir a la entidad Origen interactuar con el Receptor en todas las fases del protocolo principal.
- Identificar al usuario de la aplicación Web, así como el entorno desde el que accede dicho usuario. Con el objetivo de conocer en todo momento que usuario de la aplicación (Origen) ha iniciado una determinada ejecución y desde que entorno ejecuta cada uno de los pasos, permitiendo o no continuar al usuario con la ejecución dependiendo si se encuentra en el entorno adecuado o no.
- Gestionar las ejecuciones del protocolo principal, registrando los pasos que se han ejecutado y los resultados obtenidos en cada uno de ellos.

- Implementar la lógica de la entidad Receptor en cuanto a generación de firmas electrónicas basadas en extSP; así como validación de firmas electrónicas comprobando que cumplen con los requisitos dentro de la transacción establecidos en la extSP, comprobando en todo caso que cumplen las reglas de las SP. Para ello bede acceder a los servicios del PSC, tanto para la consulta de SP y extSP, como para realizar consultas de estados de revocación de certificados (OCSP y CRL).

## 3.7.2 Implementación

Se implementa como un módulo Web independiente al que, que una vez desplegado, pueda acceder un usuario, este usuario representa la entidad Origen del protocolo. En los siguientes apartados se describe como se ha abordado la implementación, dentro de este proyecto Web se han implementado las clases principales incluidas en el diagrama de la *Figura 29.*

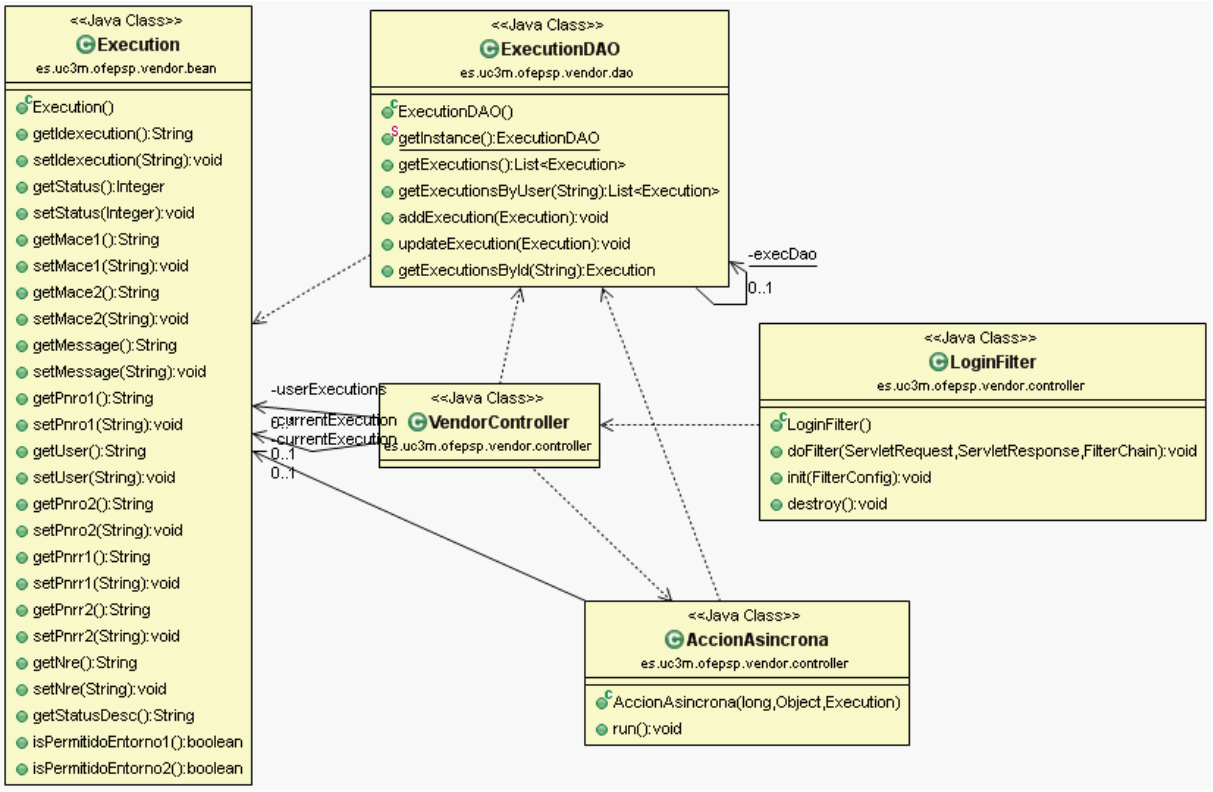



*Figura 29: Diagrama de clases del módulo Web del Receptor*

### *3.7.2.1Modelo de datos y persistencia*

La aplicación WEB dispone de un base de datos tipo [SQLite] que permite almacenar la información de las ejecuciones del protocolo en curso o ya finalizadas. Esta información queda registrada en la siguiente tabla de la Base de Datos de esta aplicación Web:

| **TABLA EXECUTION**<br>Tabla en la que se almacenan las ejecuciones iniciadas | | |
|---|---|---|
| **CAMPO** | **TIPO** | **DESCRIPCION** |
| IDEXECUTION | CHAR(40) | Identificador único de la ejecución |
| USER | CHAR(20) | Nombre del usuario (Origen) del módulo Web del receptor |
| IDSTATUS | INTEGER | Identificador del estado de la ejecución |
| MACE1 | CHAR(200) | Identificación del entorno OE1 |
| MACE2 | CHAR(200) | Identificación del entorno OE2 |
| MESSAGE | TEXT | Datos intercambiados en la ejecución |
| E_PNRO1 | TEXT | Contenido de la evidencia PNRO1 generada |
| E_PNRR1 | TEXT | Contenido de la evidencia PNRR1 generada |
| E_PNRO2 | TEXT | Contenido de la evidencia PNRO2 generada |
| E_PNRR2 | TEXT | Contenido de la evidencia PNRR2 generada |
| E_NRE | TEXT | Contenido de la evidencia NRE generada |

El módulo Web, usa el ORM *Hibernate* para realizar las operaciones de alta, baja y peticiones de consulta necesarias sobre esta tabla. Para ello se implementa una clase que permite representa en objetos planos Java (POJOs) la información de la tabla:

```
es.uc3m.ofepsp.web_vendor.bean.Execution ↔ Tabla EXECUTION
```

Para la realización de operaciones sobre la la base de datos, se implementa un DAO en la clase *ExecutionDAO* con las operaciones necesarias para obtener, modificar y dar de alta información de ejecuciones del protocolo. Este DAO será usado únicamente por el controlador *VendorController* para la gestión de ejecuciones iniciadas desde su aplicación Web.

### 3.7.2.2Publicación e interacción con aplicación cliente

Este módulo Web debe publicar la aplicación cliente para su descarga y ejecución por el Origen. Para ello se incluye el Applet implementado en el módulo *ofepspplus_buyerApplet* dentro de los recursos a desplegar en la aplicación Web.

La comunicación entre este Applet y el servidor del Receptor, será posible por el despliegue de funciones JavaScript que se ejecutan en el navegador del Origen. Estas funciones JavaScript se encuentran implementadas en el fichero *localBuyer.js* incluido en el módulo Web. Las funciones JavaScript se encargan de invocar los métodos del Applet, obteniendo los resultados e incluyéndolos en campos de formularios cuyos valores son recepcionados por la aplicación Web.

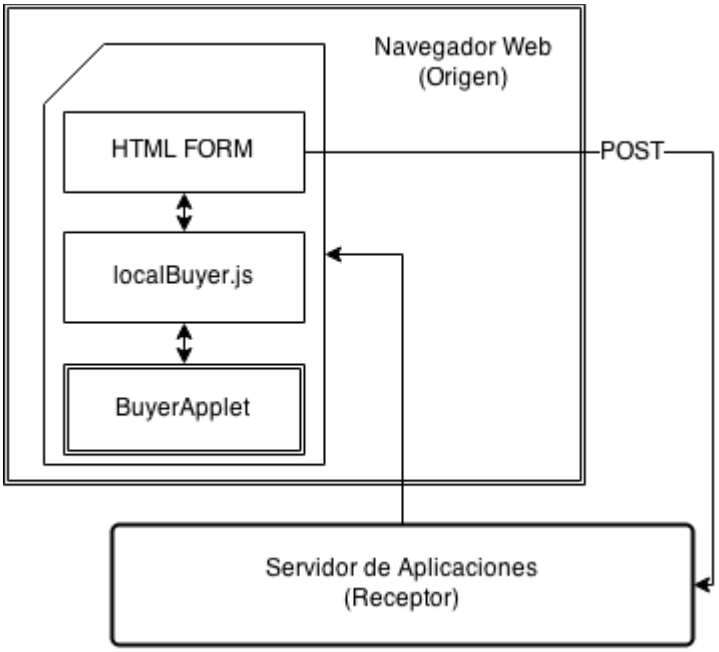



*Figura 30: Envío de evidencias del Origen a Receptor*

En *localBuyer.js* se implementa una función por cada evidencia a generar por el Origen: *generatePNRO1(), generatePNRO2(), generateNRE()*. Estas funciones invocan a las funciones análogas del Applet, obteniendo los parámetros necesarios del formulario cargado en el navegador Web que contiene tanto campos a introducir por el Origen (por ejemplo ruta a PKCS#12) como campos ocultos con valores proporcionados por el Receptor (por ejemplo PNRR1). Las propias funciones JavaScript modifican los valores del formulario con los resultados obtenidos del Applet, de tal modo que al enviarse el formulario al Receptor este obtenga las evidencias generadas por el Origen.

#### *3.7.2.3* Identificación de usuarios en sesión

Cada vez que un usuario realiza una navegación u otro tipo de petición a la aplicación Web, el módulo Web es capaz de ver si dicho usuario se ha identificado en la aplicación comprobando si ha indicado su nombre, y en caso contrario muestra la página para introducirlo. Esto se realiza implementando un

filtro configurado en el web.xml de la aplicación Web. Este filtro, implementado en la clase *LoginFilter,* comprueba que existe un objeto en sesión con el nombre del usuario, sin realizar ningún tipo de autenticación.

En los casos que en sesión no exista un objeto con el nombre del usuario se solicita dicho nombre, mostrando la vista *loginPage.xhtml*. En esta página se compone de un formulario con un único campo y la acción de login implementada en el método *onLogin()* existente en la clase *VendorController.* Este método añadirá el nombre introducido en la sesión y realizará la navegación a la vista de gestión de las ejecuciones del usuario identificado.

#### *3.7.2.4*Gestión de ejecuciones en curso de un Origen

Una vez identificado un usuario en la aplicación, se muestra la vista *vendorPage.xhtml*. Desde esta página se permite ejecutar la acción de inicio de una nueva ejecución, a la vez que se muestra un listado de las ejecuciones en curso y finalizadas del Origen que se ha identificado.

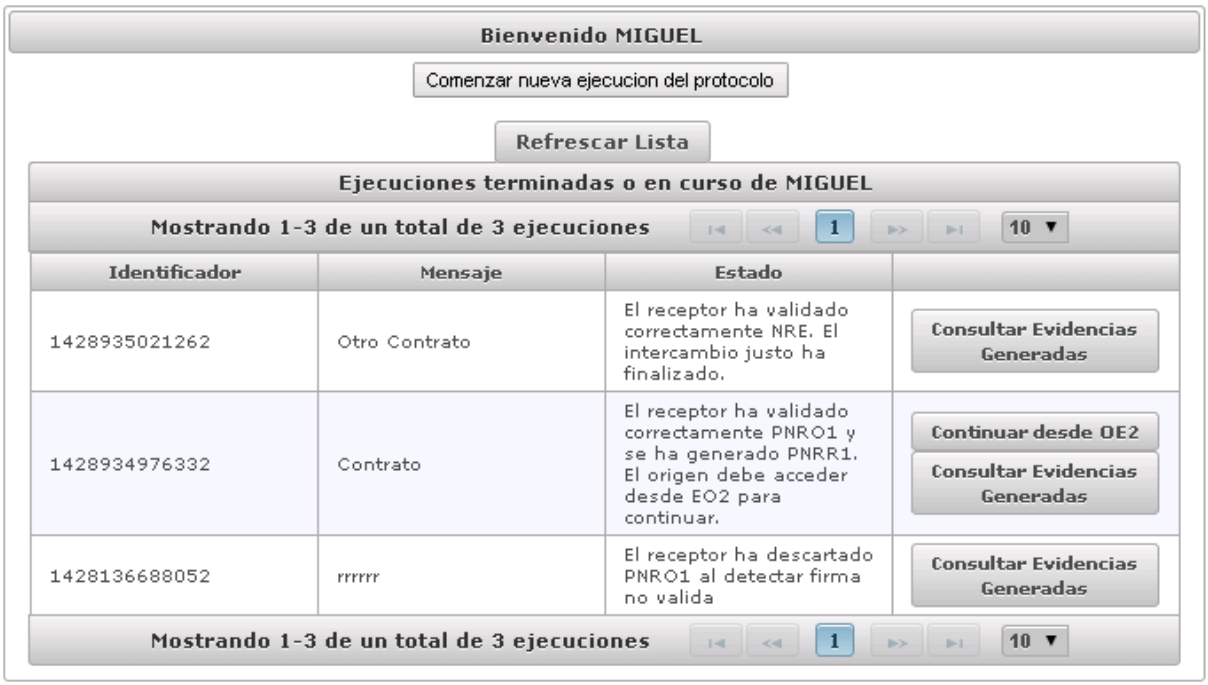



*Figura 31: Gestión de ejecuciones en aplicación del Receptor*

De cada ejecución se muestra la siguiente información y opciones de acción por parte del Origen:

- Identificador único de la ejecución.
- Mensaje, con los datos intercambiados.
- Descripción del estado de la ejecución, se distinguen los siguientes posibles:
  - El Origen ha generado PNRO1, pero aun no ha sido validado por el receptor

- El receptor ha descartado PNRO1 al detectar firma no valida
- El receptor ha validado correctamente PNRO1, pero aun no ha generado PNRR1
- El receptor ha validado correctamente PNRO1, pero ocurrió algún error en receptor generar PNRR1
- El receptor ha validado correctamente PNRO1 y se ha generado PNRR1. El Origen debe acceder desde EO2 para continuar
- El Origen ha generado PNRO2, pero aun no ha sido validado por el receptor
- El receptor ha descartado PNRO2 al detectar firma no valida. El Origen puede entrar desde OE2 para volver a generar PNRO2
- El receptor ha validado correctamente PNRO2, pero aun no ha generado PNRR2
- El receptor ha validado correctamente PNRO2, pero ocurrió algún error en receptor generar PNRR2
- El receptor ha validado correctamente PNRO2 y se ha generado PNRR2. El Origen debe acceder desde EO1 para continuar.
- El receptor ha descartado NRE al detectar firma no valida. El Origen puede entrar desde OE1 para volver a generar NRE.
- El receptor ha validado correctamente NRE. El intercambio justo ha finalizado.

- Se permite consultar el contenido de las evidencias generadas hasta el momento tanto por el Origen como por el Receptor durante la ejecución.


*Figura 32: Consulta de las evidencias generadas en una ejecución en curso o finalizada*

- Por cada ejecución se da la opción de ejecutar las acciones para continuar con dicha ejecución por parte del Origen, solo se muestran las acciones permitidas en cada entorno:

- Continuar desde OE1 para realizar PNRO2
- Continuar desde OE2 para realizar NRE

La obtención de la lista de ejecuciones y su información se encuentra implementado en el controlador *VendorController*, particularmente el método  permite obtener dicha lista. En el mismo controlador se encuentran implementadas las acciones accesibles desde la vista *vendorPage.xhtml* para iniciar o continuar una ejecución del protocolo.

### *3.7.2.5*Inicio de ejecución y recepción de PNRO1

Un determinado usuario identificado, puede iniciar una ejecución del protocolo. Esta acción muestra la vista *startPage.xhtml* que se compone de un formulario en el que se puede introducir la información necesaria para generar PNRO1. Dentro de *startPage.xhtml* se crea una instancia del Applet para permitir al usuario ejecutar la generación de PNRO1.

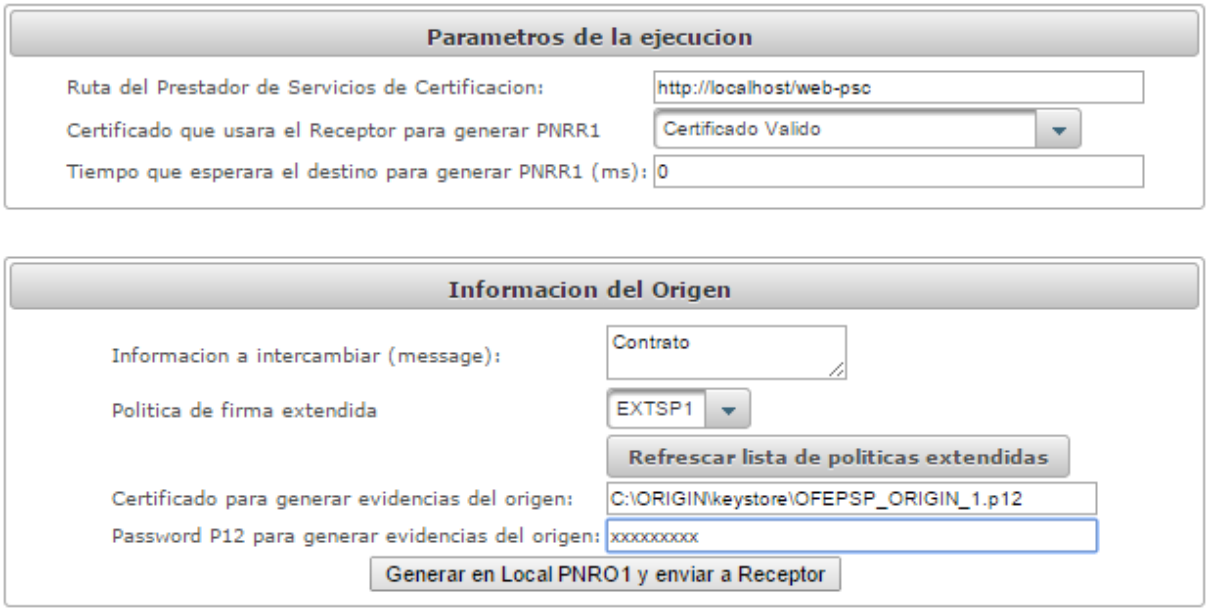



*Figura 33: Formulario para la generación de PNRO1*

El formulario existente en *startPage.xhtml* permite indicar tanto información obligatoria del Origen, como parámetros que indican el comportamiento que se quiere que tengan las entidades en la ejecución:

- Información necesaria del Origen para generar PNRO1:
  - Información a intercambiar, corresponde al mensaje que se intercambia en la ejecución del protocolo.

  - Política de Firma Extendida, identificador de la extSP que se usará durante toda la ejecución del protocolo para obtener los árboles de solución y reglas de las SP que apliquen.
  - Ruta la PKCS#12 que contenga el certificado firmante con el que generar PNRO1.
  - Contraseña del PKCS#12.
- Parámetros de la ejecución, no son necesarios por el Origen, pero nos permiten simular distintos casos de prueba.
  - Ruta del PSC que tenga desplegados los servicios de validación a usar (CRL/OCSP), así como los servicios de publicación de SP y extSP.
  - Certificado que usará la aplicación Web para generar la evidencia del Receptor PNRR1, se puede indicar si se desea que el receptor genere la firma con un certificado válido, un certificado revocado existente en la CRL, o un certificado revocado que no está aún en la CRL.
  - Tiempo que tardará la aplicación Web en comenzar a generar PNRR1 desde que se valide correctamente PNRO1. Estableciendo tiempos elevados se puede comprobar el comportamiento cuando no se cumplen con las condiciones de extSP por parte del Receptor.

La lógica de negocio implementada en *VendorController* se encargará de proporcionar la lista de extSP publicadas por el PSC indicado para que el usuario (Origen) seleccione aquella con la que desea ejecutar el protocolo. Una vez introducidos todos los datos, el Origen podrá generar PNRO1 al ejecutar la función JavaScript correspondiente, y enviando PNRO1 a la aplicación Web (Receptor) por medio del formulario existente en la vista.

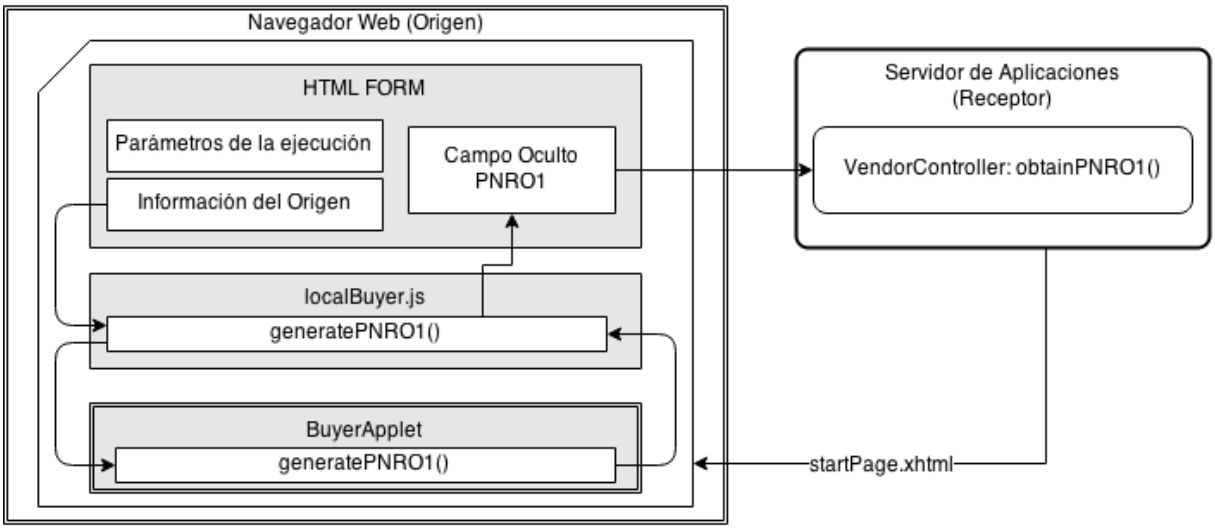



*Figura 34: Generación y envío de PNRO1*

La recepción de PNRO1 en el servidor del Receptor se implementa en el método *obtainPNRO1()* existente en *VendorController*. Este método valida PNRO1 y genera PNRR1 con la configuración indicada en el inicio del protocolo realizando las siguientes acciones:

- Se obtiene del Origen el contenido de PNRO1 y el identificador de ejecución asignado por el Origen a la nueva ejecución del protocolo. Adicionalmente se obtienen los parámetros de ejecución indicados en el formulario.
- Se almacena en la base de datos los datos obtenidos creando una nueva entrada en la tabla EXECUTION.
- Se valida PNRO1 usando la acción implementada en el módulo *ofepspplus_signature*.
    - Si la validación no es correcta se muestra en la misma página la información obtenida del error de validación para que el Origen vuelva a generar PNRO1.
    - Si la validación de PNRO1 es satisfactoria se genera PNRR1 de forma asíncrona esperando el tiempo indicado en los parámetros de la ejecución. Una vez validado correctamente PNRO1 se muestra al usuario la información del inicio de la ejecución con la vista *endStartPage.xml* avisando al Origen que es necesario que acceda desde otro entorno (OE2) para continuar la ejecución del protocolo.

#### 3.7.2.6Generación y recepción de PNRO2

En la vista *vendorPage.xml* el Origen puede ejecutar la acción de continuar una ejecución del protocolo en la que se haya generado PNRR1, pero que aún no se haya generado PNRO2. Para generar PNRO2, la aplicación Web proporciona al Origen un formulario implementado en la vista *generatePNRO2Page.xhtml* en la cual se crea una instancia del Applet para permitir al usuario ejecutar la generación de PNRO2. Dicho formulario contiene el valor de PNRR1 en un campo oculto para que lo pueda obtener el Origen.

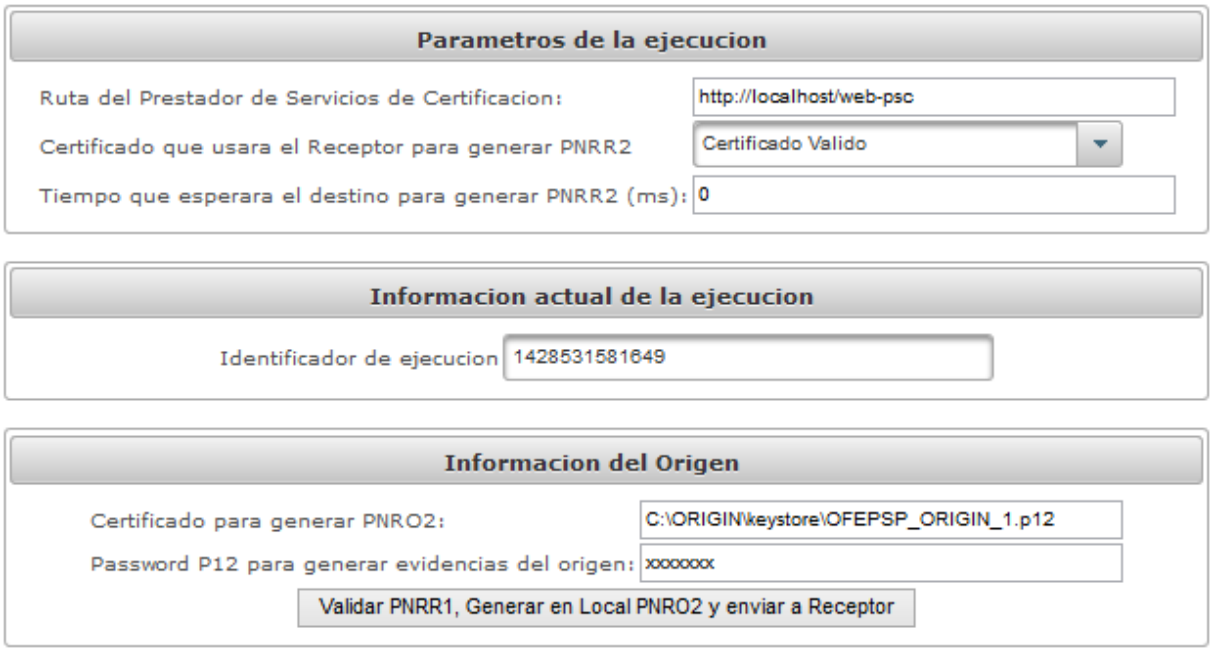



*Figura 35: Formulario para la generación de PNRO2*

El formulario existente en *generatePnro2Page.xhtml* permite indicar tanto información obligatoria del Origen, como parámetros que indican el comportamiento que se quiere que tengan las entidades en los siguientes pasos de la ejecución del protocolo, permitiendo indicar:

- Información necesaria del Origen para generar PNRO2:
  - Ruta la PKCS#12 que contenga el certificado firmante con el que generar PNRO2.
  - Contraseña del PKCS#12.
- Parámetros de la ejecución, no son necesarios ser indicados por el Origen, pero nos permiten simular distintos casos de prueba.
  - Ruta del PSC que tenga desplegados los servicios de validación a usar (CRL/OCSP), así como los servicios de publicación de SP y extSP.
  - Certificado que usará el Receptor para generar PNRR2, se puede indicar si se desea que el Receptor genere la firma con un certificado válido, un certificado revocado existente en la CRL, o un certificado revocado que no está aún en la CRL.
  - Tiempo que tardará el la aplicación Web en comenzar a generar PNRR2 desde que valide PNRO2. Estableciendo tiempos elevados se puede comprobar el comportamiento cuando no se cumplen con las condiciones de extSP por parte del Receptor.

Una vez introducidos todos los datos, el usuario de la aplicación Web (Origen) podrá generar PNRO2 al ejecutar la función JavaScript correspondiente, y enviando PNRO2 al servidor de la aplicación Web (Receptor) por medio del formulario existente en la vista.

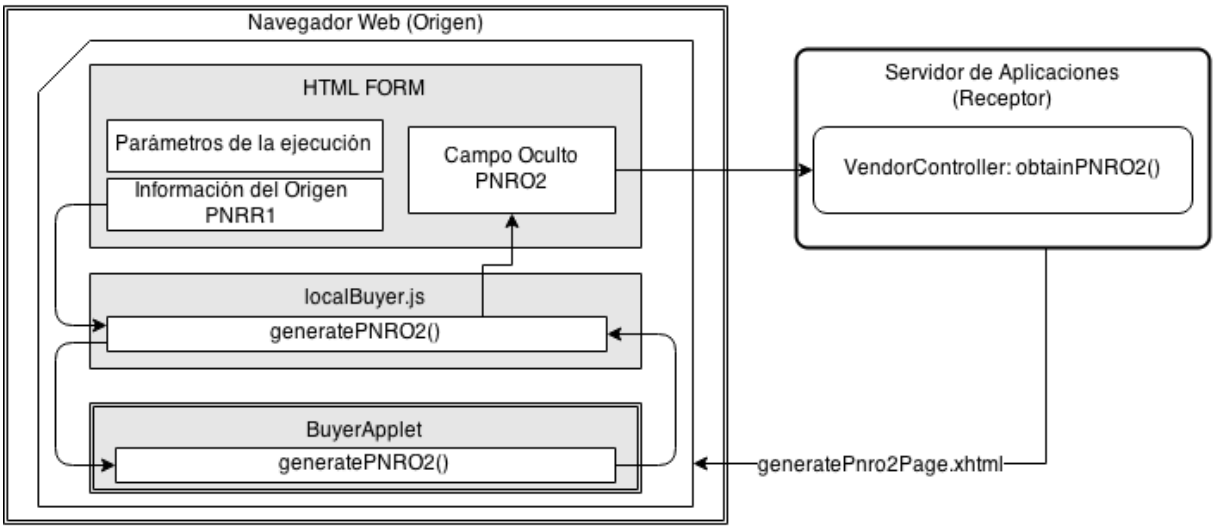



*Figura 36: Generación y envío de PNRO2*

La recepción de PNRO2 en el servidor del Receptor se implementa en el método *obtainPNRO2()* existente en *VendorController*. Este método que se ejecuta en el servidor en el que se encuentra desplegada la aplicación Web, valida PNRO2 y genera PNRR2 con la configuración indicada realizando las siguientes acciones:

- Se obtiene del Origen el contenido de PNRO2. Adicionalmente se obtienen los parámetros de ejecución indicados en el formulario.
- Se valida PNRO2 usando la acción implementada en el módulo *ofepspplus_signature*.
  - Si la validación de PNRO2 es satisfactoria se genera PNRR2 de forma asíncrona esperando el tiempo indicado en los parámetros de la ejecución. Una vez validado PNRO2 se muestra al usuario la información del inicio de la ejecución con la vista *endGeneratePnro2Page.xml* avisando al usuario (Origen) que es necesario que acceda desde el entorno inicial (OE1) para finalizar la ejecución del protocolo.
  - Si la validación no es correcta se muestra en la misma página la información obtenida del error de validación para que el Origen vuelva a generar PNRO2.

### 3.7.2.7Generación y recepción de NRE

En la vista *vendorPage.xml* el usuario de la aplicación (Origen) puede ejecutar la acción de continuar una ejecución del protocolo en la que se haya generado PNRR2, pero que aún no se haya validado correctamente NRE. Para generar NRE, el módulo Web proporciona al usuario (Origen) un formulario implementado en la vista *generateNrePage.xhtml* en la cual se crea una instancia del Applet para permitir al usuario ejecutar la generación de NRE.

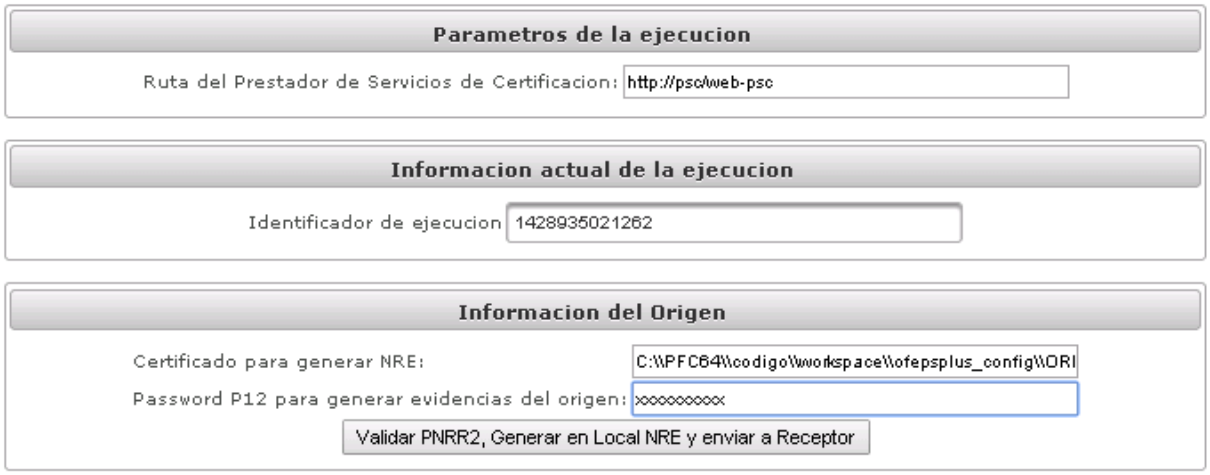



*Figura 37: Formulario para la generación de NRE*

El formulario existente en la vista *generateNrePage.xhtml* permite indicar al Origen la información necesaria para generar NRE, así como la ruta del PSC a utilizar en el resto de pasos del protocolo:

- Información necesaria del Origen para generar NRE:
  - Ruta la PKCS#12 que contenga el certificado firmante con el que generar PNRO2.
  - Contraseña del PKCS#12.
- Ruta del PSC que tenga desplegados los servicios de validación a usar (CRL/OCSP), así como los servicios de publicación de SP y extSP.

Una vez introducidos todos los datos, el usuario (Origen) podrá generar NRE al ejecutar la función JavaScript correspondiente, y enviando NRE al servidor del Receptor por medio del formulario existente en la vista.

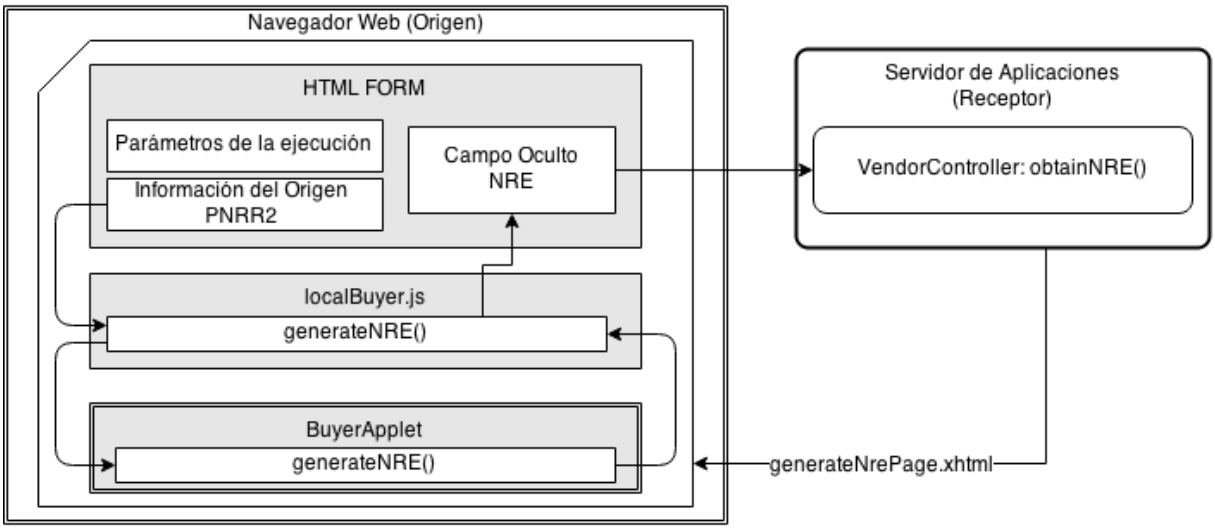



*Figura 38: Generación y envío de NRE*

La recepción de NRE en el servidor del Receptor se implementa en el método *obtainNRE()* existente en *VendorController*. Este método valida NRE finalizando la ejecución del protocolo, en este método se realizan las siguientes acciones en servidor:

- Se obtiene del Origen el contenido de NRE. Adicionalmente se obtiene la ruta del PSC indicado en el formulario de generación de NRE.
- Se valida NRE usando la acción implementada en el módulo *ofepspplus_signature*.
  - Si la validación de NRE es satisfactoria se muestra al usuario la vista *endGenerateNrePage.xml* avisando al usuario (Origen) que finalizó correctamente la ejecución del protocolo de intercambio justo.

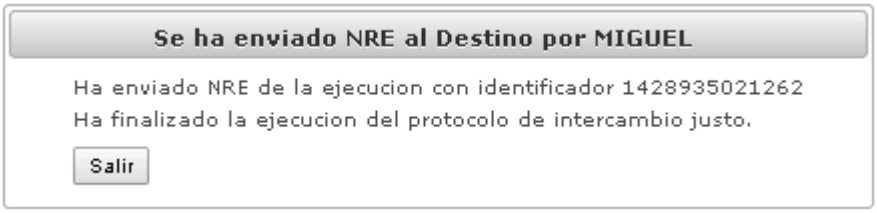



*Figura 39: Página con información de finalización del intercambio*

  - Si la validación no es correcta se muestra en la misma página la información obtenida del error de validación para que el Origen vuelva a generar NRE.

#### 3.7.2.8Generación asíncrona de evidencias

Como se ha comentado en los apartados anteriores, es posible indicar a la aplicación Web que genere las evidencias del Receptor (PNRR1 y/o PNRR2) después de un tiempo determinado, con el fin de comprobar el comportamiento cuando el Receptor no cumple las condiciones de tiempos establecidas en extSP.

Para ello, cuando en el servidor del Receptor se va a generar una evidencia, no se instancian directamente las acciones de generación del módulo *ofepspplus_signature;* si no que utiliza la acción implementada en la clase *AccionAsincrona.* Esta clase será la encargada de instanciar la acción de generación y ejecutarla en un nuevo hilo de ejecución independiente capaz de esperar el tiempo que se haya indicado.

## 3.8 *ofepspplus_ttp*, aplicación Web del TTP

### 3.8.1 Análisis funcional

Este módulo Web debe proporcionar la funcionalidad necesaria de la TTP que interviene en caso de una ejecución anómala del protocolo, sirviendo las peticiones realizadas por los usuarios de la aplicación. El usuario de la aplicación puede tratarse de las entidades Origen o Receptor dependiendo de si se solicita la ejecución del subprotocolo de interrupción o recuperación respectivamente.

- Debe permitir al usuario (Origen) realizar una solicitud firmada de interrupción de una ejecución determinada del protocolo. Ante esta petición, la aplicación Web (TTP) debe ser capaz de validar la solicitud y gestionar aquellas ejecuciones que han sido interrumpidas por solicitud del Origen. Para ello deberá contrafirmar las solicitudes de interrupción válidas y custodiar dichas evidencias.

- Del mismo modo, debe servir peticiones de recuperación; permitiendo al usuario (Receptor) en envió de las evidencias PNRO1, PNRR1, PNRO2, PNRR2 y tras su validación generar y custodiar NRE_TTE.

- En caso de haberse ejecutado el protocolo de interrupción, ante una solicitud de recuperación de la misma ejecución debe poderse recuperar y devolverse al solicitante (Receptor) la evidencia de interrupción custodiada por la TTP.

- En caso de haberse ejecutado el protocolo de recuperación, ante una solicitud de interrupción de la misma ejecución, debe poderse recuperar y devolverse al solicitante (Origen) la evidencia NRE_TTP custodiada por el TTP.

## 3.8.2 Implementación

Se implementa como una aplicación Web independiente a la que, una vez desplegado, pueda acceder un usuario (Origen o Receptor). En los siguientes apartados se describe como se ha abordado la implementación de esta aplicación Web en la que se han implementado las clases principales incluidas en el diagrama de la *Figura 40.*

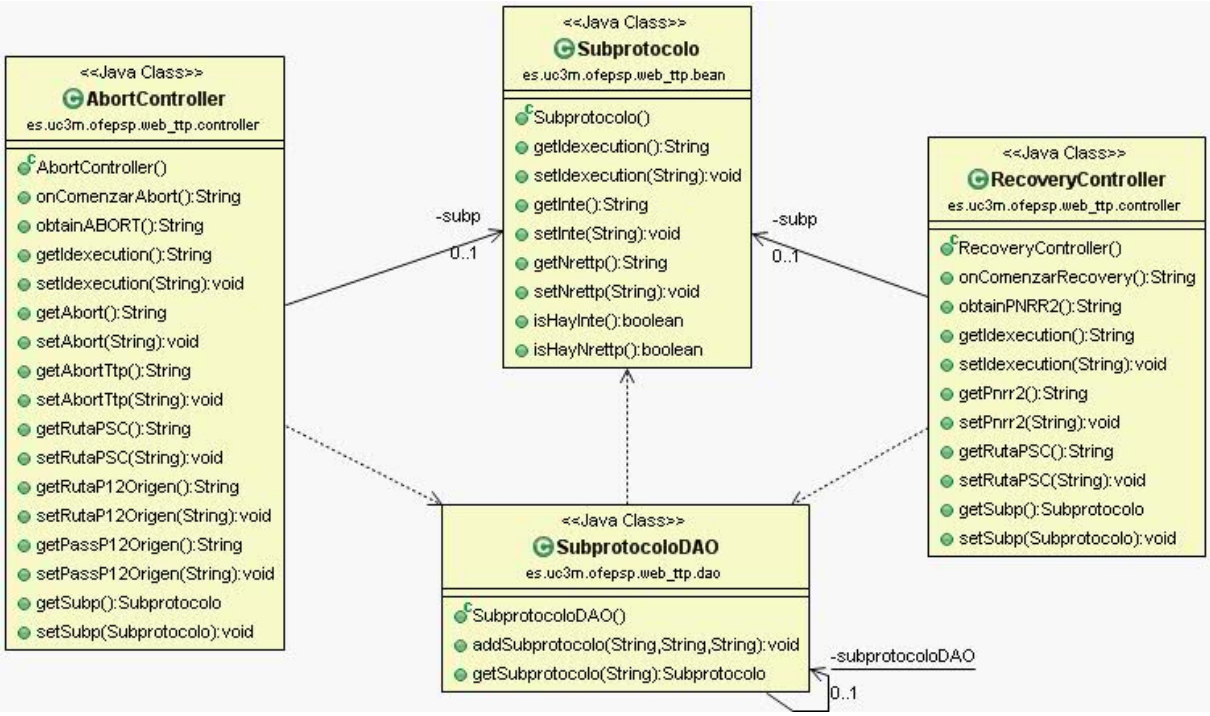



*Figura 40: Diagrama de clases del TTP*

La aplicación Web del TTP se compone de una página principal desde la que es posible acceder al inicio de ejecución de cada uno de los subprotocolos. Para cada uno de los protocolos existen dos vistas que muestran el resultado de la ejecución en el momento de finalización del subprotocolo.

### *3.8.2.1Modelo de datos y persistencia*

La aplicación Web dispone de un base de datos tipo [SQLite] con las tabla necesaria para la custodia de las evidencias generadas por TTP. Para ello de crea la tabla SUBPROTOCOLO.

| **TABLA SUBPROTOCOLO**<br>Tabla en la que se almacenan las evidencias generadas por TTP durante la ejecución de los subrpotocolos de interrupción y recuperación. | | |
|---|---|---|
| **CAMPO** | **TIPO** | **DESCRIPCION** |
| IDEXECUTION | CHAR(40) | Identificador único de la ejecución |
| E_INTE | TEXT | Contenido de la evidencia de interrupción generada por TTP al contrafirmar la solicitud del Origen |
| E_NRETTP | TEXT | Contenido de la evidencia NRE_TTP generada por TTP ante una solicitud del Receptor |

Tanto el campo E_INTE como E_NRETTP pueden tener un valor nulo. La existencia de valores en E_INTE ó E_NRTTP indica que se ejecutó el subprotocolo de interrupción o el de recuperación respectivamente para la ejecución indicada en IDEXECUTION.

Se usa el ORM *Hibernate* para realizar las operaciones de alta y peticiones de consulta necesarias sobre esta tabla. Para ello se implementa una clase que permite representa en objetos planos Java (POJOs) la información de la tabla:

```
es.uc3m.ofepsp.web_ttp.bean.Subprotocolo ↔ Tabla SUBPROTOCOLO
```

Para la realización de operaciones sobre la la base de datos, se implementa un DAO en la clase *SubprotocoloDAO* con las operaciones necesarias para obtener y dar de alta información de ejecuciones de los subprotocolos. Este DAO será usado por los controladores *AbortController* y *RecoveryController* para la gestión de evidencias generadas por TTP en las ejecuciones de subprotocolos solicitadas en su aplicación Web.

### *3.8.2.2Implementación de subprotocolo de interrupción*

La vista *abortInicio.xhtml* permite al usuario (Origen) generar una petición de interrupción firmada. El formulario existente en la vista *abortInicio.xhtml* permite indicar al Origen la información necesaria para generar dicha petición firmada, así como la ruta del PSC a utilizar en el resto de pasos del protocolo:

- Información necesaria del Origen para generar la petición firmada:
  - Ruta la PKCS#12 que contenga el certificado firmante con el que generar la firma.
  - Contraseña del PKCS#12.
- Ruta del PSC que tenga desplegados los servicios de validación a usar por la aplicación Web (TTP) en el proceso de validación.


*Figura 41: Formulario para la generación de una petición de interrupción*

Para que el usuario de la aplicación Web pueda generar peticiones de interrupción firmadas, se debe publicar la aplicación cliente para su descarga y ejecución por el usuario (Origen). Para ello se incluye el Applet implementado en el módulo *ofepspplus_buyerApplet* dentro de los recursos a desplegar en la aplicación Web del TTP. La comunicación entre este Applet y el servidor del TTP en la generación de peticiones de interrupción firmadas por el Origen, será posible por el despliegue de funciones JavaScript que se ejecutan en el navegador del Origen.

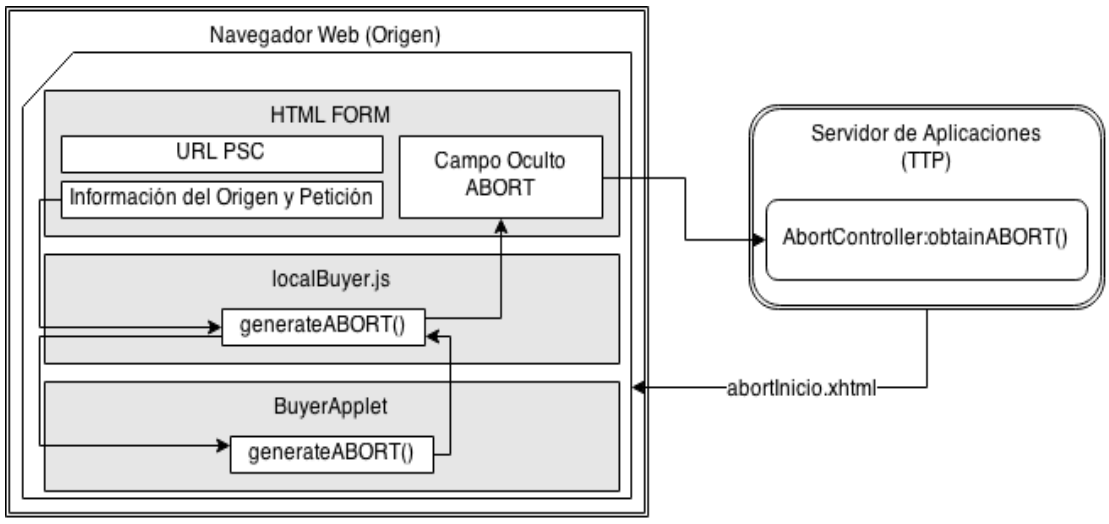



*Figura 42: Generación y envío de petición firmada del Origen a TTP*

En *localBuyer.js* se implementa la función *generateABORT()* que invoca a la función análoga del Applet, obteniendo los parámetros necesarios del formulario cargado en el navegador Web que contiene los datos a introducir por el Origen (ruta y contraseña de PKCS#12). La propia función JavaScript modifica los valores del formulario con los resultados obtenidos del Applet, de tal modo

que al enviarse el formulario al servidor del TTP este obtenga la firma generadas por el Origen con el Applet.

La recepción de la petición firmada en el servidor del TTP se implementa en el método *obtainABORT()* existente en *AbortController*. Este método validar la firma de la petición enviada por el Origen y contrafirma dicha firma usando las acciones implementadas en *ofepsplus_signature*:

- Se obtiene del Origen el contenido de la petición firmada (ABORT). Adicionalmente se obtiene la ruta del PSC indicado en el formulario de generación de la petición.
- Se obtiene de ABORT el identificador de la ejecución a interrumpir, y se consulta en la BBDD la existencia de una ejecución previa de subprotocolos para el identificador obtenido. En caso de obtenerse se muestra en la vista *abortKO.xhtml* la evidencia de interrupción o NRE-TTP custodiada por el TTP finalizando la ejecución del subprotocolo.
- Si no se obtuvo información de ninguna ejecución de subprotocolos precia, se valida ABORT usando la acción implementada en el módulo *ofepspplus_signature*.
- Si la validación de ABORT no es satisfactoria se muestra al usuario el error de validación en la misma vista.
- Si la validación de ABORT es correcta, la aplicación Web contrafirma la petición y almacena esta firma como evidencia de interrupción del TTP que muestra a su vez al usuario (Origen) en la vista *abortOK.html*.

#### *3.8.2.3Implementación de subprotocolo de recuperación*

La vista *recoverytInicio.xhtml* permite al usuario (Receptor) iniciar la ejecución del subprotocolo de recuperación. El formulario existente en la vista *recoverytInicio.xhtml* permite indicar al Receptor la información necesaria por el TTP para la generación de NRE-TTP:

- Evidencia PNRR2 que contenga la información firmada (PNRO1, PNRR1, PNRO2).
- Ruta del PSC que tenga desplegados los servicios de certificación a usar por el servidor del TTP en el proceso de validación de las evidencias.

La lógica de negocio del TTP para la ejecución del subprotocolo de recuperación se encuentra implementada en la calse *RecoveryController.* El método *obtainPNRR2()* de dicho controlador se encarga de validar la firma de las evidencias enviadas por el Receptor (PNRO1, PNRR1, PNRO2, PNRR2) y generar NRE-TTP usando las acciones implementadas en *ofepsplus_signature*:

- Se obtiene del Receptor el contenido de las evidencias generadas (PNRO1, PNRR1, PNRO2, PNRR2). Adicionalmente se obtiene la ruta del PSC indicado en el formulario.
- Se obtiene de PNRR2 el identificador de la ejecución a recuperar, y se consulta en la BBDD la existencia de una ejecución previa de subprotocolos para el identificador obtenido. En caso de obtenerse se muestra en la vista *recoveryKO.xhtml* la evidencia de interrupción o NRE-TTP custodiada por el TTP finalizando la ejecución del subprotocolo.
- Si no se obtuvo información de ninguna ejecución de subprotocolos precia, se validan las evidencias PNRR2, PNRO2, PNRR1, PNRO1 usando las acciones de validación implementadas en el módulo *ofepspplus_signature*.

- Si la validación no es satisfactoria se muestra al usuario el error de validación en la misma vista.
- Si la validación de es correcta, se genera en servidor la evidencia NRE-TTP usando la acción de generación de NRE implementada en *ofepspplus_signature* y la almacena en la base de datos de la aplicación Web del TTP, a la vez muestra NRE-TTP al usuario (Receptor) en la vista *recoveryOK.html*.

## 3.9 Despliegue y ejecución

Para la ejecución del protocolo es necesaria la generación de los módulos Web y su despliegue en servidores de aplicaciones. Para ello es necesario realizar los pasos descritos en los siguientes apartados, teniendo en cuenta que para la generación de los módulos se utilizará la herramienta [MAVEN] en su versión 2, y que dichos módulos se desplegarán en servidores de aplicaciones Tomcat7.

Para generar las aplicaciones Web necesarias para la ejecución del protocolo, inicialmente es necesario generar el módulo de firma y cliente de firma (Applet), una vez generadas estas dependencias se podrán generar los desplegables de los módulos Web.

Durante el proceso de generación de los distintos módulos la herramienta Maven descargará las dependencias de los proyectos implementados de los repositorios centrales Maven, por lo que la maquina desde la que se ejecuten los comandos de generación debe tener acceso a estos repositorios públicos [MVNCENTRAL].

### 3.9.1 Configuración de los módulos implementados

Inicialmente, y antes de generar los desplegables es necesario modificar los parámetros de configuración para el entorno en el que se vayan a desplegar los componentes necesarios.

Cada uno de los tres componentes Web dispone de una carpeta de recursos con sus PKCS#12 y base de datos SQLite. En cada una de las aplicaciones es necesario indicar la ruta en la que se encuentra dicha carpeta, para ello antes de la generación de los módulos Web es necesario modificar la ruta en la que se encuentran dichos recursos modificando los parámetros de los ficheros:

*ofepspplus_psc\src\main\resources\es\uc3m\ofepsp\web_psc\config.properties*
*ofepspplus_ttp\src\main\resources\es\uc3m\ofepsp\web_ttp\config.properties*
*ofepspplus_vendor\src\main\resources\es\uc3m\ofepsp\vendor\config.properties*

### 3.9.2 Generación del módulo de firma y cliente de firma

Para construir el componente de firma, será necesario ejecutar el comando *"mvn install"* desde la ruta raíz del proyecto *ofepspplus_psc*. Como resultado se obtendrá un fichero JAR que corresponde al componente de firma y que se podrán encontrar en el repositorio Maven local en las ruta:
*[REPOSITORIO MAVEN]/es/uc3m/ofepsp/api-signature/1.0/api-signature-1.0.jar*

Una vez generado el componente de firma se podrá ejecutar el comando *"mvn install"* desde la raíz del proyecto *ofepspplus_buyerApplet*. El resultado serán un JAR que corresponde al Applet firmado que se podrá encontrar en el repositorio Maven local en las ruta:
*[REPOSITORIO MAVEN]/es/uc3m/ofepsp/buyerApplet/1.0/buyerApplet-1.0.jar*

### 3.9.3 Generación y despliegue de los módulos Web

Previo a su generación es necesario incorporar el componente *buyerApplet-1.0.jar* en los recursos a publicar por el Receptor y TTP. Para ello se copiará dicho componente a las rutas:

*ofepspplus_ttp\src\main\webapp\faces*
*ofepspplus_vendor\src\main\webapp\xhtml*

Posteriormente, para la generación de los tres módulos es necesario ejecutar el comando *"mvn install*"desde las rutas raíz de los proyecto *ofepspplus_psc, ofepsplus_vendor, ofepspplus_ttp*. Estos comandos descargarán las dependencias de los proyectos al repositorio Maven local y generará el módulo Web (WAR) en la carpeta *tarject* de cada uno de los proyectos.

Una vez generados, se desplegarán los WAR en los servidores Web en los que se quieran publicar los servicios de cada una de las entidades.

### 3.9.4 Ejecución de protocolo

Una vez generados y desplegados los módulos Web, se podrá acceder a los servicios de cada uno de ellos para la ejecución del protocolo principal y subprotocolos. Para ello se seguirán los siguientes pasos:

1. Generar las SP/extSP desde la aplicación Web del PSC publicada.
2. Acceder a la aplicación Web del Receptor desde un entorno inicial (OE1) e iniciar la ejecución del protocolo principal indicando los datos del Origen.

3. Acceder desde el entorno OE1 u otro entorno OE2 para realizar los siguientes pasos del protocolo.

En cualquier momento se podrá acceder a la aplicación Web del TTP para iniciar la ejecución del subprotocolo de interrupción o de recuperación indicando los datos necesarios en los formularios publicados por la aplicación Web del TTP.

Durante la ejecución, tanto las aplicaciones servidor, como la aplicación cliente *(Applet)* muestran en la consola del servidor y/o en la consola Java, las trazas con información de las operaciones que se realizan en cada momento. Adicionalmente es posible consultar en la aplicación Web del Receptor, y en la aplicación del TTP; las evidencias generadas en la ejecución del protocolo principal, y las generadas en la ejecución de subprotocolos respectivamente.

# Capítulo 4. Conclusiones y trabajo futuro

## 4.1 Conclusiones

Una vez completada la implementación del protocolo OFEPSP+ para un escenario determinado, se puede decir que se ha logrado el objetivo principal del proyecto, el comprobar la viabilidad de implementación del protocolo, y demostrar que el uso de Políticas de Firmas Extendidas no es sólo útil, sino viable tecnológicamente.

Desde la publicación de la primera versión de la Política de Firma de la AGE, no han sido pocos los organismo públicos, tanto de la Administración General, como Administración Local (por ejemplo el Ayuntamiento de Madrid) que han publicado su propia Política de Firma. La definición de las Políticas de Firma Extendidas, así como la implementación de un caso de uso completo y totalmente funcional de dichas políticas en el presente proyecto, demuestran que las carencias de las Políticas de Firma existentes pueden solventarse con el uso de Políticas de Firma Extendidas; ya que estas pueden permitir a los organismos definir las reglas, no sólo en cuanto a generación y validación de firmas de modo individual, sino en transacciones completas como son los tramites en la Administración Pública.

Las transacciones en las que varias entidades deben generar varias firmas, no son exclusivas de la Administración Pública. En otros ámbitos privados, durante el intercambio de información se hace imprescindible el uso de protocolos de intercambio justo en los que podría utilizarse el protocolo OFEPSP+ si se disponen de los componentes software necesarios, y si se involucran las entidades responsables para la emisión de servicios de certificación, e entidades independientes que permitan que sea posible la resolución de disputas.

Una vez implementado y ejecutado el protocolo OFEPSP+, también se pueden ver los problemas que pueden llegar a tener las entidades involucradas, sobretodo en los casos que estas entidades se correspondan a personas físicas. En especial al Origen se le obliga a disponer dos entornos en los que pueda ejecutar componentes software con acceso a servicios de certificación, aunque este componente no fuera un Applet, se le está dificultando a esta entidad la realización del intercambio. Incluso en un entorno real lo más posible es que el Origen no pueda acceder a los servicios de certificación ya que la consulta de estados de revocación de certificados de ciertas PSC reconocidas (por ejemplo la

FNMT) son de pago y soló es posible usarlos si se contratan previamente estos servicios. Esto en muchos casos no es aceptable, sobre todo en los casos que el Origen sea un cliente potencial de un bien o servicio del Receptor.

Ejecutando el protocolo con la implementación realizada, se ve más factible la ejecución del protocolo en intercambios de información B2B, en los que los componentes software a usar por las entidades podrían ser proporcionados por un PSC reconocido que a su vez publique los servicios de certificación necesarios.

## 4.2 Trabajo futuro

### 4.2.1 Implementación para distintos modelos de negocio.

La implementación se ha realizado para un modelo B2C, en el que el Origen corresponde a una persona física que accede a una aplicación Web del Receptor. En el modelo implementado la complejidad es mayor, ya que ha sido necesario implementar aplicaciones Web así como aplicaciones clientes que interactúan entre ellas. Con una evolución de la implementación realizada, podría realizarse la implementación para simular la ejecución del protocolo en un modelo B2B.

Para el caso de comunicaciones C2C, se podría realizar un estudio mas detallado, en el que se pueda determinar la viabilidad de ejecución del protocolo con un modelo C2C. En este caso, posiblemente tendría que modificarse el comportamiento de las distintas entidades, tanto Origen, Receptor como del Tercero de Confianza.

### 4.2.2 Ampliación de funcionalidad.

A continuación se numeran varios puntos que han quedado fuera del alcance de la implementación pero que podrían incluirse en evolutivos:

- **Sellado de Tiempo**. Inclusión de servicios de sellado de tiempo, permitiendo al PSC generar y validar sellos de tiempo según las reglas existentes en las SP. El uso de sellos de tiempo, también implicaría implementar en el módulo de firma la funcionalidad necesaria para generar firmas con formatos que admitan sellados de tiempo y realizar sus validaciones.

- **Mayor interpretación de las políticas**. La implementación actual comprueba los requisitos básicos de las firmas en su generación y validación (periodo de firma, certificados admitidos, métodos de validación permitidos, algoritmos a usar, etc.). Pero han quedado fuera de la implementación otras reglas que podrían incluirse en las Políticas de Firma.

- La implementación actual se ha centrado en los formatos basados en XML, las Políticas de Firma y las Políticas de Firma Extendidas se publican y utilizan en formato XML, así como las firmas realizadas para la generación de evidencias son a su vez formatos de firma XML (XAdES). Podría realizarse una evolución para **permitir formatos ASN.1**, en el que las evidencias se basen en firmas ASN.1 (CAdES).

- En la implementación realizada se ha acotado el número de algoritmos usados, tanto de transformación como de cálculo de hash y cifrado, teniendo en cuenta los mas comunes y aquellos proporcionados por el proveedor criptográfico usado. Con la **inclusión de proveedores criptográficos más avanzados**, o realizando nuevas implementaciones, pueden incluirse mayor número y más robustos algoritmos.

- **Uso de listas de servicios de certificación**. En la actualidad el uso de Políticas de Firma y de listas de servicios de certificación esta sumamente ligado. Una lista de servicios de certificación [TSL], permite numerar los proveedores de servicios de certificación y sus servicios reconocidos. Al incluir la referencia de una TSL en una Política de Firma, se pueden establecer los servicios admitidos para dicha política: tipos de certificados admitidos en cada momento, métodos de validación permitidos, servicios de sellado que pueden usarse en el ámbito de la política, etc. Por ello, una evolución de la implementación, e incluso del propio protocolo OFEPSP+, se podría realizar mediante el estudio y uso de TSL.

# Bibliografía y referencias

[JLHA] Jorge López hernández-Ardieta. Enhancing the reliability of digital signatures as non-repudiation evidence under a holistic threat model . Febrero 2011

[RFC3280] RFC 3280 - Internet X.509 Public Key Infrastructure, Certificate and Certificate Revocation List (CRL) Profile. Abril 2002.

[RFC3161] RFC 3161 - Internet X.509 Public Key Infrastructure, TimeStamp Protocol. Agosto 2001.

[POLAGE] Información y descargas sobre la política de firma de la Administración General del Estado: http://administracionelectronica.gob.es/ctt/politicafirma

[TR102041] ETSI TR 102 041 v1.1.1 - Signature Policies Report. Febrero 2002

[TR102271] ETSI TR 102 271 v1.1.1 - ASN.1 format for signature policies. Diciembre 2003

[TR102038] ETSI TR 102 038 v1.1.1 - XML format for signature policies. Abril 2002

[POLFE] Información y descargas de la política de firma de Facturae, http://www.facturae.gob.es/formato/Paginas/politicas-firma-electronica.aspx

[XCA] http://sourceforge.net/projects/xca, herramienta para administración de claves asimétricas como RSA o DSA . Permite simular una pequeña CA . Utiliza la biblioteca OpenSSL para las operaciones criptográficas .

[SQLite] http://www.sqlite.org, base de datos relacional ligera.

[MAVEN] http://maven.apache.org, herramienta de gestión de dependencias y generación de desplegables

[XADES132] ETSI TS 101 903 V1.3.2 - XML Advanced Electronic Signatures (XadES). Marzo 2006

[JSR222] Java Specification Requests - Java Architecture for XML Binding (JAXB) 2.0. Agosto 2010

[PAGNIA] H. Pagnia, F.C. Gartner. On the impossibility of fair exchange without a trusted third party. Technical Report, Febrero 1999

[OFEP] N. Asokan, M. Schunter and M. Waidner. Optimistic protocols for fair exchange; 4th ACM Conference on Computer and Communications Security, pág. 7-17. 1997

[MICALI] N. Asokan, V. Shoup and M. Waidner: Asynchronous Protocols for Optimistic Fair Exchange; IEEE Symposium on Research in Security and Privacy. 1998.

[FPH] Bao, F., Wang, G., Zhou, J., Zhu, Z.: Analysis and Improvement of Micali's Fair Contract Signing Protocol, Proceedings of The 9th Australasian Conference on Information Security and Privacy, pág. 176-187. 2004

[TORRENT] B. Cohen. Incentives build robustness in bittorrent. International workshop on Peer-To-Peer Systems. 2003.

[JXADES] https://github.com/universitatjaumei/jxades

[XADES132] ETSI TS 101 903 V1.3.2 - XML Advanced Electronic Signatures (XAdES). Marzo 2006

[JSR105] Java XML Digital Signature API Specification (JSR 105), https://jcp.org/en/jsr/detail?id=105

[RFC2560] RFC 2560 - X.509 Internet Public Key Infrastructure. Online Certificate Status Protocol – OCSP. Junio 1999.

[RFC2253] RFC 2253 - LADPv3 Distinguished Names. Diciembre 1997.

[MVNCENTRAL] Repositorio central de librerías de código abierto http://central.sonatype.org/

[TSL] ETSI TS 102 231 V3.1.2 - Requirements for Trust Service Provider status information. Diciembre 2009.

# Anexo A. Planificación y presupuesto

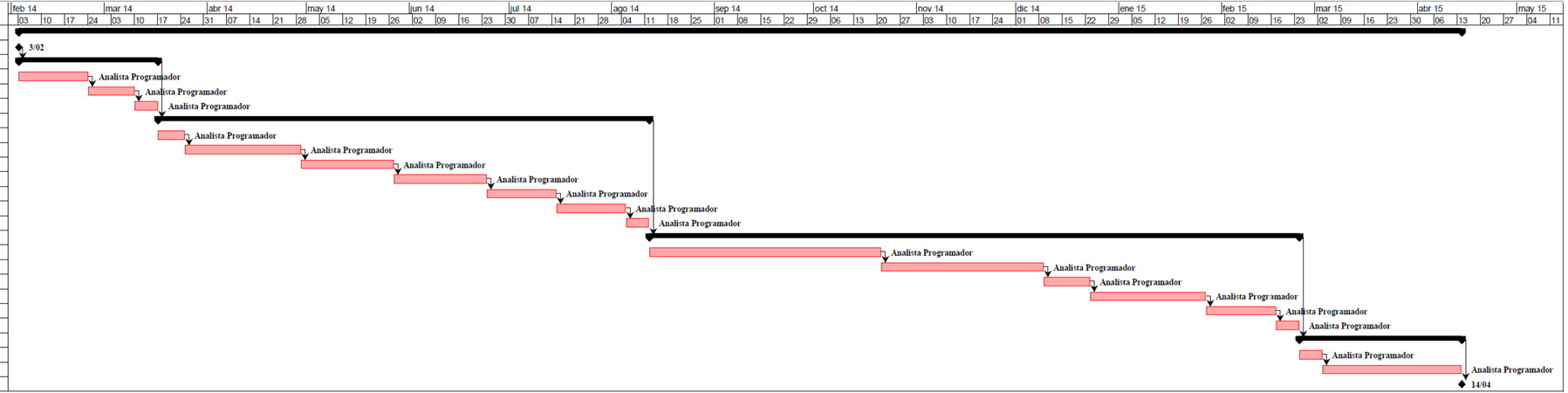
En este anexo se incluye la información referente a la planificación, y recursos necesarios en las diferentes tareas llevadas a cabo para realizar una implementación del protocolo OFEPSP+. La información obtenida nos permite conocer una estimación del tiempo total y presupuesto de implementación del protocolo OFEPSP+. La aplicación utilizada para llevar a cabo esta planificación y presupuesto del proyecto ha sido ProjectLibre (*http://www.projectlibre.org*).

## A.1. Diagrama de GANTT

A continuación se muestra el diagrama Gantt donde se pueden ver las tareas llevadas a cabo para el desarrollo de los distintos módulos necesarios para la ejecución del protocolo, así como las relaciones entre dichas tareas.

Mediante este diagrama se va a obtener una estimación del tiempo que conlleva el desarrollo y los recursos que se destinan a cada tarea. En este cálculo se ha asumido que en todas las fases de este proyecto se precisa un Ingeniero Técnico en Informática de Gestión (Analista Programador) con más de 5 años de experiencia en proyectos de firma electrónica, supervisado por un Jefe de Proyecto (Tutor/Co-director); aunque el proyecto podría abordarse con perfiles sin experiencia, el tiempo de desarrollo del sistema sería mayor, especialmente en fases de implementación.

| | Nombre | Predecesores | Duracion | Inicio | Terminado |
|---|---|---|---|---|---|
| 1 | **PROYECTO OFEPSP+** | | **311 days** | **3/02/14 9:00** | **14/04/15 9:00** |
| 2 | Inicio | | 0 days | 3/02/14 9:00 | 3/02/14 9:00 |
| 3 | **ANTEPROYECTO** | **2** | **30 days** | **3/02/14 9:00** | **17/03/14 9:00** |
| 4 | Estudio SP/extSP OFEPSP+ | | 15 days | 3/02/14 9:00 | 24/02/14 9:00 |
| 5 | Definición Alcance y Requisitos | 4 | 10 days | 24/02/14 9:00 | 10/03/14 9:00 |
| 6 | Documentación | 5 | 5 days | 10/03/14 9:00 | 17/03/14 9:00 |
| 7 | **ANÁLISIS Y DISEÑO** | **3** | **106 days** | **17/03/14 9:00** | **12/08/14 9:00** |
| 8 | Definición Arquitectura | | 6 days | 17/03/14 9:00 | 25/03/14 9:00 |
| 9 | API de firma e interpretación políticas | 8 | 25 days | 25/03/14 9:00 | 29/04/14 9:00 |
| 10 | Aplicación WEB y servicios del PSC | 9 | 20 days | 29/04/14 9:00 | 27/05/14 9:00 |
| 11 | Cliente de Firma | 10 | 20 days | 27/05/14 9:00 | 24/06/14 9:00 |
| 12 | Aplicación WEB del Receptor | 11 | 15 days | 24/06/14 9:00 | 15/07/14 9:00 |
| 13 | Aplicación WEB del TTP | 12 | 15 days | 15/07/14 9:00 | 5/08/14 9:00 |
| 14 | Documentación | 13 | 5 days | 5/08/14 9:00 | 12/08/14 9:00 |
| 15 | **IMPLEMENTACION Y PRUEBAS UNITARIAS** | **7** | **140 days** | **12/08/14 9:00** | **24/02/15 9:00** |
| 16 | API de firma e interpretación políticas | | 50 days | 12/08/14 9:00 | 21/10/14 9:00 |
| 17 | Aplicación WEB y servicios del PSC | 16 | 35 days | 21/10/14 9:00 | 9/12/14 9:00 |
| 18 | Cliente de Firma | 17 | 10 days | 9/12/14 9:00 | 23/12/14 9:00 |
| 19 | Aplicación WEB del Receptor | 18 | 25 days | 23/12/14 9:00 | 27/01/15 9:00 |
| 20 | Aplicación WEB del TTP | 19 | 15 days | 27/01/15 9:00 | 17/02/15 9:00 |
| 21 | Documentación | 20 | 5 days | 17/02/15 9:00 | 24/02/15 9:00 |
| 22 | **CIERRE DE PROYECTO** | **15** | **35 days** | **24/02/15 9:00** | **14/04/15 9:00** |
| 23 | Despliegue de aplicaciones y ejecucion | | 5 days | 24/02/15 9:00 | 3/03/15 9:00 |
| 24 | Desarrollo de documentación (Memoria) | 23 | 30 days | 3/03/15 9:00 | 14/04/15 9:00 |
| 25 | Finalización | 22 | 0 days | 14/04/15 9:00 | 14/04/15 9:00 |



# A.2. Presupuesto

En este anexo se muestra de manera detallada el presupuesto desglosado del proyecto.

## Recursos humanos

Basándonos en el diagrama de Gantt del apartado anterior se obtiene el número de horas, teniendo en cuenta que la planificación se ha realizado con un recurso disponible (Ingeniero Técnico, analista programador con más de cinco años de experiencia en firma electrónica) con jornada de 8 horas diarias. En este cálculo, se tiene en cuenta que durante las distintas fases del proyecto colabora un Jefe de Proyecto (tutor/co-director). Las tarifas incluidas de cada perfil se ajustan a proyectos similares en el mercado.

| Cargo | Nº de Horas | Coste Hora | Dedicación* | Total (€) |
|---|---|---|---|---|
| Ingeniero Técnico | 2488 | 26 | 1 | 64688 |
| Ingeniero Senior | 497 | 36 | 0,2 | 17892 |
| | | | | Coste total sin IVA: 82580 € |

*Tabla 7: Costes recursos humanos*

*Utilizando 1 Hombre/ mes = 160 horas según la planificación realizada

## Recursos hardware y software

Para la implementación y ejecución del protocolo se requiere un ordenador que permita ejecutar varios servidores de aplicaciones de forma simultanea, sin ser necesarias grandes capacidades de procesamiento pero si de memoria RAM. Por ello, durante la realización del proyecto se ha adquirido un equipo portátil con 12Gb de RAM y un procesador de gama media/baja por 399€.

A parte del Sistema Operativo y software preinstalado en el portátil adquirido, todo el software utilizado durante la realización del proyecto: entono de desarrollo, servidores de aplicaciones, APIs java, aplicaciones microinformáticas, etc. tienen licencias abiertas sin coste *(The MIT License, Apache License, GNU General Public License, Eclipse Public License, etc.).*

| Descripción | Coste (€) | % de uso dedicado | Dedicación meses | Periodo de depreciación | Coste imputable (€) |
|---|---|---|---|---|---|
| Ordenador portatil (AMD Quad-Core Processor A6-5200, 12GB DDR3 SODIMM) | 399 | 100 | 10,1 | 60 | 67 |

*Tabla 8: Costes hardware y software*

## Subcontratación de tareas y otros costes

No aplican costes de subcontratación. Tampoco costes de contratación de servicios, ya que en vez de contratar servicios de certificación se ha optado por implementar aquellos necesarios para la ejecución del protocolo.

A los costes finales se les incluirá una tasa de costes indirectos del 20% relacionados con los riesgos del proyecto y otros costes no contemplados. A parte se incluye un coste del 21% en concepto de impuestos sobre el valor añadido.

## Resumen de costes totales

| Descripción | Costes totales (€) |
|---|---|
| Recursos humanos | 82580 |
| Amortización del hardware | 67 |
| Costes de software | 0 |
| Costes indirectos (20%) | 16529,4 |
| Total antes de impuestos | 99176,4 |
| IVA (21%) | 20827,0 |
| **TOTAL** | **120003,4 €** |

*Tabla 9: Costes totales*

# Anexo B. Ejemplos de SP/extSP

## B.1. Ejemplo de Política de Firma en formato XML

A continuación se muestra un ejemplo del contenido de una SP en formato XML, generada y publicada con la aplicación Web del PSC usando la implementación del módulo *ofepspplus_psc,* e interpretable por cualquier entidad con el uso de la librería implementada en el módulo *ofepspplus_signature*:

<ns3:signaturePolicyType xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"
xmlns="http://uri.etsi.org/2038/v1.1.1#" xmlns:ns3="http://uri.etsi.org/01903/v1.3.2EXT#">

<SignPolicyDigestAlg Algorithm="http://www.w3.org/2001/04/xmlenc#sha256" />
<ns2:Transforms>
<ns2:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
</ns2:Transforms>
<SignPolicyInfo>
<SignPolicyIdentifier>
<ns3:Identifier>SP2_LOCATION</ns3:Identifier>
</SignPolicyIdentifier>
<DateOfIssue>2015-04-04T00:00:00.000+02:00</DateOfIssue>
<PolicyIssuerName>CN=OFEPSP_TSP, OU=UC3M, O=UC3M, L=Leganes,
ST=Madrid, C=ES</PolicyIssuerName>
<FieldOfApplication>OFEPSP</FieldOfApplication>
<SignatureValidationPolicy>
<SigningPeriod>
<NotBefore>2015-04-04T00:00:00.000+02:00</NotBefore>
<NotAfter>2016-04-04T00:00:00.000+02:00</NotAfter>
</SigningPeriod>
<CommonRules>
<SigningCertTrustCondition>
<SignerTrustTrees>
<CertificateTrustPoint>
<TrustPoint>MIIEJTCCAw2gAwIBAgIBATANBgkqhkiG9w0BAQsFADBkMQswCQYDVQQGEwJFUzEPM
A0GA1UECBMGTWFkcmlkMRAwDgYDVQQHEwdMZWdhbmVzMQ0wCwYDVQQKEwRVQzNNMQ0w
CwYDVQQLEwRVQzNNMRQwEgYDVQQDDAtPRkVQU1BfQ0FfMTAeFw0xNDEyMjkwNjI2MDBaFw0y

NDEyMjkwNjI2MDBaMGQxCzAJBgNVBAYTAkVTMQ8wDQYDVQQIEwZNYWRyaWQxEDAOBgNVBA
cTB0xlZ2FuZXMxDTALBgNVBAoTBFVDM00xDTALBgNVBAsTBFVDM00xFDASBgNVBAMMC09GR
VBTUF9DQV8xMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA+vJWGrzFNTcckTXkrDZrbv
imEQu64K1K0KXCJhHfBHQfPOxjeOVQgOwOZ7QEjGEqojVTMIPcdV74syT8q0i5KGx6kFoSgO5IHE4F5nf
OZ6wQpgnoAS5tCZ0Mr2WNijhnHqZ1FLySoXlF6jrSbe94cbCXbAXMje9+TbsbwQaEKNw5CW+3pddP9U7
mI6fbbxWrRYuR9iAsJVg7lBQejRsuk41Nb/vpqF00fOaYnWhNm1XWMzoWtgxnmkXfRV3uYt6Vja4ixaG/7R
3dT8ZRdqIuaoaIDaj4KT+Ed1yOP23iCmJizyVpCxb1KpAH6c9WDBc3UvXnJwUFyNd10jkmCnS3OwIDAQA
Bo4HhMIHeMAwGA1UdEwQFMAMBAf8wHQYDVR0OBBYEFA5wlY9an5+1O+43NhErskLK7uNlMIGOB
gNVHSMEgYYwgYOAFA5wlY9an5+1O+43NhErskLK7uNloWikZjBkMQswCQYDVQQGEwJFUzEPMA0G
A1UECBMGTWFkcmlkMRAwDgYDVQQHEwdMZWdhbmVzMQ0wCwYDVQQKEwRVQzNNMQ0wCwY
DVQQLEwRVQzNNMRQwEgYDVQQDDAtPRkVQU1BfQ0FfMYIBATALBgNVHQ8EBAMCAf4wEQYJY
IZIAYb4QgEBBAQDAgLEMA0GCSqGSIb3DQEBCwUAA4IBAQCFgOopvOJK8blBzqj+dmZL7iVu3zw+zd
W9uoD65unhN0kDg2/8iI1wiH3zOPJqCfWscESKucoknIJSO7QNKgx+Bp8LxA5MTZYPagf9TgZN6Ni/6Tu7D
4o6fyYVZW8NMXVi8ChisYFGCva9v5bIg3MUi0DBxYjyKSduDLCdLt3pw9D4zdRTVvR/NKZaGHuHhh+iF
LHSf1h1slZAtsekEE/riEINDH341WaZXgXE80vTLCyHGGdjXpWF/SDg7f/oCh1BYiCjfr2o/EZC4fc24Q2THf
FdKgAybD7fuzob9f4IWkvXvUypFL8kRt+GuJKPm6XidtxeZMCrf2TBdW3XnFMm

```
                                        </TrustPoint>
                                    </CertificateTrustPoint>
                                </SignerTrustTrees>
                                <SignerRevReq>
                                    <EndRevReq>
                                        <EnuRevReq>bothcheck</EnuRevReq>
                                    </EndRevReq>
                                    <CACerts>
                                        <EnuRevReq>bothcheck</EnuRevReq>
                                    </CACerts>
                                </SignerRevReq>
                            </SigningCertTrustCondition>
                            <AlgorithmConstraintSet>
                                <SignerAlgConstraints>
                                    <AlgAndLength>
                                <AlgId>http://www.w3.org/2000/09/xmldsig#rsa-sha1</AlgId>
                                    </AlgAndLength>
                                </SignerAlgConstraints>
                            </AlgorithmConstraintSet>
                        </CommonRules>
                        <CommitmentRules>
                            <CommitmentRule>
                                <SelCommitment>
                                    <SelCommitmentType>
                                        <RecognizedCommitmentType>
                                            <CommitmentIdentifier>
    <ns3:Identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin1</ns3:Identifier>
                                            </CommitmentIdentifier>
                                        </RecognizedCommitmentType>
                                    </SelCommitmentType>
                                </SelCommitment>
                                <SignerAndVerifierRules>
```

<SignerRules>
<ExternalSignedObjects>false</ExternalSignedObjects>
<MandatedSignedQProperties>
<QPropertyID>SigningTime</QPropertyID>
<QPropertyID>SigningCertificate</QPropertyID>
<QPropertyID>SignaturePolicyIdentifier</QPropertyID>
<QPropertyID>ExtSignaturePolicyIdentifier</QPropertyID>
<QPropertyID>SignerRole</QPropertyID>
<QPropertyID>CommitmentTypeIndication</QPropertyID>
<QPropertyID>SignatureProductionPlace</QPropertyID>
</MandatedSignedQProperties>
<MandatedUnsignedQProperties />
</SignerRules>
<VerifierRules />
</SignerAndVerifierRules>
<RoleTrustCondition>
<RoleMandated>true</RoleMandated>
<RoleConstraints>
<RoleValueConstraint>
<rol xmlns="" xmlns:ns4="http://uri.etsi.org/2038/v1.1.1#">Origin</rol>
</RoleValueConstraint>
</RoleConstraints>
</RoleTrustCondition>
</CommitmentRule>
<CommitmentRule>
<SelCommitment>
<SelCommitmentType>
<RecognizedCommitmentType>
<CommitmentIdentifier>
<ns3:Identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt1</ns3:Identifier>
</CommitmentIdentifier>
</RecognizedCommitmentType>
</SelCommitmentType>
</SelCommitment>
<SignerAndVerifierRules>
<SignerRules>
<ExternalSignedObjects>false</ExternalSignedObjects>
<MandatedSignedQProperties>
<QPropertyID>SigningTime</QPropertyID>
<QPropertyID>SigningCertificate</QPropertyID>
<QPropertyID>SignaturePolicyIdentifier</QPropertyID>
<QPropertyID>ExtSignaturePolicyIdentifier</QPropertyID>
<QPropertyID>SignerRole</QPropertyID>
<QPropertyID>CommitmentTypeIndication</QPropertyID>
</MandatedSignedQProperties>
<MandatedUnsignedQProperties />
</SignerRules>
<VerifierRules />

```
                </SignerAndVerifierRules>
                <RoleTrustCondition>
                        <RoleMandated>true</RoleMandated>
                        <RoleConstraints>
                                <RoleValueConstraint>
<rol xmlns="" xmlns:ns4="http://uri.etsi.org/2038/v1.1.1#">Receiver</rol>
                                </RoleValueConstraint>
                        </RoleConstraints>
                </RoleTrustCondition>
</CommitmentRule>
<CommitmentRule>
        <SelCommitment>
                <SelCommitmentType>
                        <RecognizedCommitmentType>
                                <CommitmentIdentifier>
<ns3:Identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin2</ns3:Identifier>
                                </CommitmentIdentifier>
                        </RecognizedCommitmentType>
                </SelCommitmentType>
        </SelCommitment>
        <SignerAndVerifierRules>
                <SignerRules>
<ExternalSignedObjects>false</ExternalSignedObjects>
<MandatedSignedQProperties>
        <QPropertyID>SigningTime</QPropertyID>
        <QPropertyID>SigningCertificate</QPropertyID>
        <QPropertyID>SignaturePolicyIdentifier</QPropertyID>
        <QPropertyID>ExtSignaturePolicyIdentifier</QPropertyID>
        <QPropertyID>SignerRole</QPropertyID>
        <QPropertyID>CommitmentTypeIndication</QPropertyID>
</MandatedSignedQProperties>
<MandatedUnsignedQProperties />
                </SignerRules>
                <VerifierRules />
        </SignerAndVerifierRules>
        <RoleTrustCondition>
                <RoleMandated>true</RoleMandated>
                <RoleConstraints>
                        <RoleValueConstraint>
<rol xmlns="" xmlns:ns4="http://uri.etsi.org/2038/v1.1.1#">Origin</rol>
                        </RoleValueConstraint>
                </RoleConstraints>
        </RoleTrustCondition>
</CommitmentRule>
<CommitmentRule>
        <SelCommitment>
                <SelCommitmentType>
                        <RecognizedCommitmentType>
```

```
                                                    <CommitmentIdentifier>
<ns3:Identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt2</ns3:Identifier>
                                                    </CommitmentIdentifier>
                                                </RecognizedCommitmentType>
                                            </SelCommitmentType>
                                        </SelCommitment>
                                        <SignerAndVerifierRules>
                                            <SignerRules>
                                <ExternalSignedObjects>false</ExternalSignedObjects>
                                <MandatedSignedQProperties>
                                        <QPropertyID>SigningTime</QPropertyID>
                                        <QPropertyID>SigningCertificate</QPropertyID>
                                        <QPropertyID>SignaturePolicyIdentifier</QPropertyID>
                                        <QPropertyID>ExtSignaturePolicyIdentifier</QPropertyID>
                                        <QPropertyID>SignerRole</QPropertyID>
                                        <QPropertyID>CommitmentTypeIndication</QPropertyID>
                                </MandatedSignedQProperties>
                                <MandatedUnsignedQProperties />
                                            </SignerRules>
                                            <VerifierRules />
                                        </SignerAndVerifierRules>
                                        <RoleTrustCondition>
                                            <RoleMandated>true</RoleMandated>
                                            <RoleConstraints>
                                                <RoleValueConstraint>
                                <rol xmlns="" xmlns:ns4="http://uri.etsi.org/2038/v1.1.1#">Receiver</rol>
                                                </RoleValueConstraint>
                                            </RoleConstraints>
                                        </RoleTrustCondition>
                                    </CommitmentRule>
                                    <CommitmentRule>
                                        <SelCommitment>
                                            <SelCommitmentType>
                                                <RecognizedCommitmentType>
                                                    <CommitmentIdentifier>
<ns3:Identifier>http://uri.etsi.org/01903/v1.2.2#NonRepudiationEvidence</ns3:Identifier>
                                                    </CommitmentIdentifier>
                                                </RecognizedCommitmentType>
                                            </SelCommitmentType>
                                        </SelCommitment>
                                        <SignerAndVerifierRules>
                                            <SignerRules>
                                <ExternalSignedObjects>false</ExternalSignedObjects>
                                <MandatedSignedQProperties>
                                        <QPropertyID>SigningTime</QPropertyID>
                                        <QPropertyID>SigningCertificate</QPropertyID>
                                        <QPropertyID>SignaturePolicyIdentifier</QPropertyID>
                                        <QPropertyID>ExtSignaturePolicyIdentifier</QPropertyID>
```

```
                                        <QPropertyID>SignerRole</QPropertyID>
                                        <QPropertyID>CommitmentTypeIndication</QPropertyID>
                                    </MandatedSignedQProperties>
                                    <MandatedUnsignedQProperties />
                                            </SignerRules>
                                            <VerifierRules />
                                        </SignerAndVerifierRules>
                                        <RoleTrustCondition>
                                            <RoleMandated>true</RoleMandated>
                                            <RoleConstraints>
                                                    <RoleValueConstraint>
                                    <rol xmlns="" xmlns:ns4="http://uri.etsi.org/2038/v1.1.1#">Origin</rol>
                                                    </RoleValueConstraint>
                                                    <RoleValueConstraint>
                                    <rol xmlns="" xmlns:ns4="http://uri.etsi.org/2038/v1.1.1#">TTP</rol>
                                                    </RoleValueConstraint>
                                            </RoleConstraints>
                                        </RoleTrustCondition>
                                    </CommitmentRule>
                                </CommitmentRules>
                            </SignatureValidationPolicy>
                    </SignPolicyInfo>
<SignPolicyDigest>
                    ekU0QmRPL0pLNnJwN1VTKzhKc2VIT1hELzRlZDA1aTFUc1Qwa1pJc1pPVT0=
</SignPolicyDigest>
</ns3:signaturePolicyType>
```

# B.2. Ejemplo de Política de Firma Extendida en formato XML

A continuación se muestra un ejemplo del contenido de una extSP en formato XML, generada y publicada con la aplicación Web del PSC usando la implementación del módulo *ofepspplus_psc,* e interpretable por cualquier entidad con el uso de la librería implementada en el módulo *ofepspplus_signature*:

```
<ns2:extSignaturePolicy xmlns:ns2="http://jlopez.thesis.uc3m.es/ETS-ExtendedElectronicSignaturePolicies-97Syntax">
  <ns2:extSignPolicyInfo>
    <ns2:extSignPolicyIdentifier>
      <ns2:extSignPolicyId>EXTTOS</ns2:extSignPolicyId>
      <ns2:dateOfIssue>2015-04-08T00:00:00.000+02:00</ns2:dateOfIssue>
    </ns2:extSignPolicyIdentifier>
    <ns2:extSignValidationPolicy>
      <ns2:signingPeriod>
        <ns2:notBefore>2015-04-08T00:00:00.000+02:00</ns2:notBefore>
        <ns2:notAfter>2016-04-08T00:00:00.000+02:00</ns2:notAfter>
      </ns2:signingPeriod>
      <ns2:treesOfSolutions>
        <ns2:treeOfSignature>
          <ns2:signature>
            <ns2:identifier>1</ns2:identifier>
            <ns2:signer>10</ns2:signer>
            <ns2:acceptableSignPolicies>SP1</ns2:acceptableSignPolicies>
            <ns2:allowedCommitmentTypes>
              <ns2:CHOICE>
                <ns2:recognizedCommitmentType>
<ns2:identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin1</ns2:identifier>
                  <ns2:fieldOfApplication>
                    <ns2:printableString>First partial non-repudiation of origin</ns2:printableString>
                  </ns2:fieldOfApplication>
                </ns2:recognizedCommitmentType>
              </ns2:CHOICE>
            </ns2:allowedCommitmentTypes>
          </ns2:signature>
          <ns2:signature>
            <ns2:identifier>2</ns2:identifier>
            <ns2:signer>20</ns2:signer>
            <ns2:acceptableSignPolicies>SP1</ns2:acceptableSignPolicies>
            <ns2:allowedCommitmentTypes>
              <ns2:CHOICE>
```

```
                    <ns2:recognizedCommitmentType>
<ns2:identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt1</ns2:identifier>
                        <ns2:fieldOfApplication>
                            <ns2:printableString>First partial non-repudiation of receipt</ns2:printableString>
                        </ns2:fieldOfApplication>
                    </ns2:recognizedCommitmentType>
                </ns2:CHOICE>
            </ns2:allowedCommitmentTypes>
            <ns2:counterSignatures>
                <ns2:signature>
                    <ns2:identifier>3</ns2:identifier>
                    <ns2:signer>10</ns2:signer>
                    <ns2:acceptableSignPolicies>SP1</ns2:acceptableSignPolicies>
                    <ns2:allowedCommitmentTypes>
                        <ns2:CHOICE>
                            <ns2:recognizedCommitmentType>
<ns2:identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin2</ns2:identifier>
                                <ns2:fieldOfApplication>
                                <ns2:printableString>Second partial non-repudiation of origin</ns2:printableString>
                                </ns2:fieldOfApplication>
                            </ns2:recognizedCommitmentType>
                        </ns2:CHOICE>
                    </ns2:allowedCommitmentTypes>
                    <ns2:counterSignatures>
                        <ns2:signature>
                            <ns2:identifier>4</ns2:identifier>
                            <ns2:signer>20</ns2:signer>
                            <ns2:acceptableSignPolicies>SP1</ns2:acceptableSignPolicies>
                            <ns2:allowedCommitmentTypes>
                                <ns2:CHOICE>
                                    <ns2:recognizedCommitmentType>
<ns2:identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt2</ns2:identifier>
                                        <ns2:fieldOfApplication>
                                <ns2:printableString>Second partial non-repudiation of receipt</ns2:printableString>
                                        </ns2:fieldOfApplication>
                                    </ns2:recognizedCommitmentType>
                                </ns2:CHOICE>
                            </ns2:allowedCommitmentTypes>
                            <ns2:counterSignatures>
                                <ns2:signature>
                                    <ns2:identifier>5</ns2:identifier>
                                    <ns2:signer>10</ns2:signer>
                                    <ns2:acceptableSignPolicies>SP1</ns2:acceptableSignPolicies>
                                    <ns2:allowedCommitmentTypes>
                                        <ns2:CHOICE>
                                            <ns2:recognizedCommitmentType>
<ns2:identifier>http://uri.etsi.org/01903/v1.2.2#NonRepudiationEvidence</ns2:identifier>
```

```
                        <ns2:fieldOfApplication>
                           <ns2:printableString>Non-repudiation evidence</ns2:printableString>
                        </ns2:fieldOfApplication>
                     </ns2:recognizedCommitmentType>
                  </ns2:CHOICE>
               </ns2:allowedCommitmentTypes>
               <ns2:timingAndSequence>
                  <ns2:relativeTimingAndSequence>
                     <ns2:RelativeTimingAndSequence>
                        <ns2:pathToRefSignature>2 3 4</ns2:pathToRefSignature>
                        <ns2:maxDelta>
                           <ns2:deltaSeconds>0</ns2:deltaSeconds>
                           <ns2:deltaMinutes>15</ns2:deltaMinutes>
                           <ns2:deltaHours>0</ns2:deltaHours>
                           <ns2:deltaDays>0</ns2:deltaDays>
                        </ns2:maxDelta>
                     </ns2:RelativeTimingAndSequence>
                  </ns2:relativeTimingAndSequence>
               </ns2:timingAndSequence>
            </ns2:signature>
         </ns2:counterSignatures>
         <ns2:timingAndSequence>
            <ns2:relativeTimingAndSequence>
               <ns2:RelativeTimingAndSequence>
                  <ns2:pathToRefSignature>2 3</ns2:pathToRefSignature>
                  <ns2:maxDelta>
                     <ns2:deltaSeconds>0</ns2:deltaSeconds>
                     <ns2:deltaMinutes>15</ns2:deltaMinutes>
                     <ns2:deltaHours>0</ns2:deltaHours>
                     <ns2:deltaDays>0</ns2:deltaDays>
                  </ns2:maxDelta>
               </ns2:RelativeTimingAndSequence>
            </ns2:relativeTimingAndSequence>
         </ns2:timingAndSequence>
      </ns2:signature>
   </ns2:counterSignatures>
   <ns2:timingAndSequence>
      <ns2:relativeTimingAndSequence>
         <ns2:RelativeTimingAndSequence>
            <ns2:pathToRefSignature>2</ns2:pathToRefSignature>
            <ns2:maxDelta>
               <ns2:deltaSeconds>0</ns2:deltaSeconds>
               <ns2:deltaMinutes>15</ns2:deltaMinutes>
               <ns2:deltaHours>0</ns2:deltaHours>
               <ns2:deltaDays>0</ns2:deltaDays>
            </ns2:maxDelta>
         </ns2:RelativeTimingAndSequence>
      </ns2:relativeTimingAndSequence>
```

```
                    </ns2:timingAndSequence>
                  </ns2:signature>
                </ns2:counterSignatures>
                <ns2:timingAndSequence>
                  <ns2:relativeTimingAndSequence>
                    <ns2:RelativeTimingAndSequence>
                      <ns2:pathToRefSignature>1</ns2:pathToRefSignature>
                      <ns2:maxDelta>
                        <ns2:deltaSeconds>0</ns2:deltaSeconds>
                        <ns2:deltaMinutes>15</ns2:deltaMinutes>
                        <ns2:deltaHours>0</ns2:deltaHours>
                        <ns2:deltaDays>0</ns2:deltaDays>
                      </ns2:maxDelta>
                    </ns2:RelativeTimingAndSequence>
                  </ns2:relativeTimingAndSequence>
                </ns2:timingAndSequence>
              </ns2:signature>
            </ns2:treeOfSignature>
            <ns2:treeOfSignature>
              <ns2:signature>
                <ns2:identifier>1</ns2:identifier>
                <ns2:signer>10</ns2:signer>
                <ns2:acceptableSignPolicies>SP1</ns2:acceptableSignPolicies>
                <ns2:allowedCommitmentTypes>
                  <ns2:CHOICE>
                    <ns2:recognizedCommitmentType>
<ns2:identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin1</ns2:identifier>
                      <ns2:fieldOfApplication>
                        <ns2:printableString>First partial non-repudiation of origin</ns2:printableString>
                      </ns2:fieldOfApplication>
                    </ns2:recognizedCommitmentType>
                  </ns2:CHOICE>
                </ns2:allowedCommitmentTypes>
              </ns2:signature>
              <ns2:signature>
                <ns2:identifier>2</ns2:identifier>
                <ns2:signer>20</ns2:signer>
                <ns2:acceptableSignPolicies>SP1</ns2:acceptableSignPolicies>
                <ns2:allowedCommitmentTypes>
                  <ns2:CHOICE>
                    <ns2:recognizedCommitmentType>
<ns2:identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt1</ns2:identifier>
                      <ns2:fieldOfApplication>
                        <ns2:printableString>First partial non-repudiation of receipt</ns2:printableString>
                      </ns2:fieldOfApplication>
                    </ns2:recognizedCommitmentType>
                  </ns2:CHOICE>
                </ns2:allowedCommitmentTypes>
```

```
            <ns2:counterSignatures>
              <ns2:signature>
                <ns2:identifier>3</ns2:identifier>
                <ns2:signer>10</ns2:signer>
                <ns2:acceptableSignPolicies>SP1</ns2:acceptableSignPolicies>
                <ns2:allowedCommitmentTypes>
                  <ns2:CHOICE>
                    <ns2:recognizedCommitmentType>
<ns2:identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin2</ns2:identifier>
                      <ns2:fieldOfApplication>
                       <ns2:printableString>Second partial non-repudiation of origin</ns2:printableString>
                      </ns2:fieldOfApplication>
                    </ns2:recognizedCommitmentType>
                  </ns2:CHOICE>
                </ns2:allowedCommitmentTypes>
                <ns2:counterSignatures>
                  <ns2:signature>
                    <ns2:identifier>4</ns2:identifier>
                    <ns2:signer>20</ns2:signer>
                    <ns2:acceptableSignPolicies>SP1</ns2:acceptableSignPolicies>
                    <ns2:allowedCommitmentTypes>
                      <ns2:CHOICE>
                        <ns2:recognizedCommitmentType>
<ns2:identifier>http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt2</ns2:identifier>
                          <ns2:fieldOfApplication>
                     <ns2:printableString>Second partial non-repudiation of receipt</ns2:printableString>
                          </ns2:fieldOfApplication>
                        </ns2:recognizedCommitmentType>
                      </ns2:CHOICE>
                    </ns2:allowedCommitmentTypes>
                    <ns2:counterSignatures>
                      <ns2:signature>
                        <ns2:identifier>5</ns2:identifier>
                        <ns2:signer>10</ns2:signer>
                        <ns2:acceptableSignPolicies>SP2_LOCATION</ns2:acceptableSignPolicies>
                        <ns2:allowedCommitmentTypes>
                          <ns2:CHOICE>
                            <ns2:recognizedCommitmentType>
<ns2:identifier>http://uri.etsi.org/01903/v1.2.2#NonRepudiationEvidence</ns2:identifier>
                              <ns2:fieldOfApplication>
                                <ns2:printableString>Non-repudiation evidence</ns2:printableString>
                              </ns2:fieldOfApplication>
                            </ns2:recognizedCommitmentType>
                          </ns2:CHOICE>
                        </ns2:allowedCommitmentTypes>
                        <ns2:timingAndSequence>
                          <ns2:relativeTimingAndSequence>
                            <ns2:RelativeTimingAndSequence>
```

```
                        <ns2:pathToRefSignature>2 3 4</ns2:pathToRefSignature>
                        <ns2:maxDelta>
                          <ns2:deltaSeconds>0</ns2:deltaSeconds>
                          <ns2:deltaMinutes>15</ns2:deltaMinutes>
                          <ns2:deltaHours>0</ns2:deltaHours>
                          <ns2:deltaDays>0</ns2:deltaDays>
                        </ns2:maxDelta>
                      </ns2:RelativeTimingAndSequence>
                    </ns2:relativeTimingAndSequence>
                  </ns2:timingAndSequence>
                </ns2:signature>
              </ns2:counterSignatures>
              <ns2:timingAndSequence>
                <ns2:relativeTimingAndSequence>
                  <ns2:RelativeTimingAndSequence>
                    <ns2:pathToRefSignature>2 3</ns2:pathToRefSignature>
                    <ns2:maxDelta>
                      <ns2:deltaSeconds>0</ns2:deltaSeconds>
                      <ns2:deltaMinutes>15</ns2:deltaMinutes>
                      <ns2:deltaHours>0</ns2:deltaHours>
                      <ns2:deltaDays>0</ns2:deltaDays>
                    </ns2:maxDelta>
                  </ns2:RelativeTimingAndSequence>
                </ns2:relativeTimingAndSequence>
              </ns2:timingAndSequence>
            </ns2:signature>
          </ns2:counterSignatures>
          <ns2:timingAndSequence>
            <ns2:relativeTimingAndSequence>
              <ns2:RelativeTimingAndSequence>
                <ns2:pathToRefSignature>2</ns2:pathToRefSignature>
                <ns2:maxDelta>
                  <ns2:deltaSeconds>0</ns2:deltaSeconds>
                  <ns2:deltaMinutes>15</ns2:deltaMinutes>
                  <ns2:deltaHours>0</ns2:deltaHours>
                  <ns2:deltaDays>0</ns2:deltaDays>
                </ns2:maxDelta>
              </ns2:RelativeTimingAndSequence>
            </ns2:relativeTimingAndSequence>
          </ns2:timingAndSequence>
        </ns2:signature>
      </ns2:counterSignatures>
      <ns2:timingAndSequence>
        <ns2:relativeTimingAndSequence>
        <ns2:RelativeTimingAndSequence>
          <ns2:pathToRefSignature>1</ns2:pathToRefSignature>
          <ns2:maxDelta>
            <ns2:deltaSeconds>0</ns2:deltaSeconds>
```

```
                        <ns2:deltaMinutes>15</ns2:deltaMinutes>
                        <ns2:deltaHours>0</ns2:deltaHours>
                        <ns2:deltaDays>0</ns2:deltaDays>
                      </ns2:maxDelta>
                    </ns2:RelativeTimingAndSequence>
                  </ns2:relativeTimingAndSequence>
                </ns2:timingAndSequence>
              </ns2:signature>
            </ns2:treeOfSignature>
          </ns2:treesOfSolutions>
        </ns2:extSignValidationPolicy>
        <ns2:extSignContext>
          <ns2:businessApplicationDomain>
            <ns2:sigPolicyQualifierId>1 2 840 113549 1 9 16 5 1</ns2:sigPolicyQualifierId>
            <ns2:sigQualifier>
              <printableString>Sale of goods/international trade transactions</printableString>
            </ns2:sigQualifier>
          </ns2:businessApplicationDomain>
          <ns2:transactionalContext>
            <ns2:sigPolicyQualifierId>1 2 840 113549 1 9 16 5 1</ns2:sigPolicyQualifierId>
            <ns2:sigQualifier>
              <printableString>Purchase Order/Acceptance in relation to a book purchase order made through
Alice Bookshop Internet Web page between Alice Bookshop and a client of Alice Bookshop</printableString>
            </ns2:sigQualifier>
          </ns2:transactionalContext>
          <ns2:disputeResolution>
            <ns2:sigPolicyQualifierId>1 2 840 113549 1 9 16 5 1</ns2:sigPolicyQualifierId>
            <ns2:sigQualifier>
              <printableString>Any disputes arising under this policy shall be referred to a suitably qualified
expert, whose decision shall be final and binding upon the parties, provided that this signature policy imposes the
constraints under which any signature created under it shall be valid. The dispute resolution procedure shall be
carried out in a European court with appropriate responsibilities</printableString>
            </ns2:sigQualifier>
          </ns2:disputeResolution>
          <ns2:audienceConditions>
            <ns2:sigPolicyQualifierId>1 2 840 113549 1 9 16 5 1</ns2:sigPolicyQualifierId>
            <ns2:sigQualifier>
              <printableString>The digital signature-based evidence is only valid in a specified jurisdiction, where
laws exist which recognize the legal validity of signatures created under conditions as specified in the
policy</printableString>
            </ns2:sigQualifier>
          </ns2:audienceConditions>
        </ns2:extSignContext>
        <ns2:extSignPolExtensions>
          <ns2:SignPolExtn>
            <ns2:extnID>Transformation</ns2:extnID>
<ns2:extnValue>687474703A2F2F7777772E77332E6F72672F54522F323030312F5245432D786D6C2D63313
46E2D3230303130333135</ns2:extnValue>
```

```
            </ns2:SignPolExtn>
        </ns2:extSignPolExtensions>
    </ns2:extSignPolicyInfo>
    <ns2:extSignPolicyProtection>
        <ns2:protectionAlg>
            <ns2:algorithm>http://www.w3.org/2001/04/xmlenc#sha256</ns2:algorithm>
        </ns2:protectionAlg>
        <ns2:protection>7W5sS4O8Y293uHI7ACaDSBbGvWVxaz+gOQFikzYPYvY=</ns2:protection>
    </ns2:extSignPolicyProtection>
</ns2:extSignaturePolicy>
```

# Anexo C. Evidencias generadas

A continuación se incluye el contenido de una evidencia NRE generada en la finalización de la ejecución del protocolo principal. Esta evidencia contiene los datos e identificador de ejecución codificados en Base64, así como todas las firmas simples y *counterSignatures* generadas en la ejecución del protocolo. Aunque para incluir esta información en la memoria no se incluyen ciertos valores como partes publicas de certificados y valores de firma, se puede observar con este ejemplo la jerarquía de firmas en la evidencia de no repudio.

```
<OFEPSP>
<DATOS_INTERCAMBIADOS
        Encoding="base64" Id="DATOS_INTERCAMBIADOS-67d7ae1e-5661-4195-b7f0-8593f8c727cc">
        SURFSkVDVUNJT049MTQyODkzNTAyMTI2MiNNRU5TQUpFPU90cm8gQ29udHJhdG8=
</DATOS_INTERCAMBIADOS>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        Id="Signature-f52ee67c-929a-470c-a68a-9745e83f1f8f-Signature">
        <ds:SignedInfo>
        [...]
        <ds:Reference Id="Reference-4ed3e4fc-8dd6-4bb2-a4ea-05b1c3f70192"
        URI="#DATOS_INTERCAMBIADOS-67d7ae1e-5661-4195-b7f0-8593f8c727cc">
        [...]
        </ds:Reference>
        [...]
        </ds:SignedInfo>
        <ds:SignatureValue Id="Signature-f52ee67c-929a-470c-a68a-9745e83f1f8f-SignatureValue">
        [...]
        </ds:SignatureValue>
        <ds:KeyInfo Id="Signature-f52ee67c-929a-470c-a68a-9745e83f1f8f-KeyInfo">
        [...]
        </ds:KeyInfo>
        <ds:Object>
        <xades:QualifyingProperties xmlns:xades="http://uri.etsi.org/01903/v1.3.2EXT#"
                Id="Signature-f52ee67c-929a-470c-a68a-9745e83f1f8f-QualifyingProperties"
                Target="#Signature-f52ee67c-929a-470c-a68a-9745e83f1f8f-Signature"
                xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
```

```
        <xades:SignedProperties
        Id="Signature-f52ee67c-929a-470c-a68a-9745e83f1f8f-SignedProperties">
        <xades:SignedSignatureProperties>
        [...]
        </xades:SignedSignatureProperties>
        <xades:SignedDataObjectProperties>
        [...]
        <xades:CommitmentTypeIndication>
        <xades:CommitmentTypeId>
        <xades:Identifier Qualifier="">
        http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin1
        </xades:Identifier>
        <xades:Description />
        </xades:CommitmentTypeId>
        <xades:ObjectReference />
        <xades:CommitmentTypeQualifiers />
        </xades:CommitmentTypeIndication>
        </xades:SignedDataObjectProperties>
        </xades:SignedProperties>
        <xades:UnsignedProperties>
        <xades:UnsignedSignatureProperties />
        </xades:UnsignedProperties>
        </xades:QualifyingProperties>
        </ds:Object>
</ds:Signature>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        Id="Signature-7207fe45-6f65-46ec-84e8-636254469ef2-Signature">
        <ds:SignedInfo>
        [...]
        <ds:Reference Id="Reference-929ed41c-a9cf-4217-bd4d-b501547ab52b"
        URI="#DATOS_INTERCAMBIADOS-67d7ae1e-5661-4195-b7f0-8593f8c727cc">
        [...]
        </ds:Reference>
        [...]
        </ds:SignedInfo>
        <ds:SignatureValue Id="Signature-7207fe45-6f65-46ec-84e8-636254469ef2-SignatureValue">
        [...]
        </ds:SignatureValue>
        <ds:KeyInfo Id="Signature-7207fe45-6f65-46ec-84e8-636254469ef2-KeyInfo">
        [...]
        </ds:KeyInfo>
        <ds:Object>
        <xades:QualifyingProperties xmlns:xades="http://uri.etsi.org/01903/v1.3.2EXT#"
                Id="Signature-7207fe45-6f65-46ec-84e8-636254469ef2-QualifyingProperties"
                Target="#Signature-7207fe45-6f65-46ec-84e8-636254469ef2-Signature"
                xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <xades:SignedProperties
                Id="Signature-7207fe45-6f65-46ec-84e8-636254469ef2-SignedProperties">
```

```
<xades:SignedSignatureProperties>
[...]
</xades:SignedSignatureProperties>
<xades:SignedDataObjectProperties>
[...]
<xades:CommitmentTypeIndication>
<xades:CommitmentTypeId>
<xades:Identifier Qualifier="">
http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt1
</xades:Identifier>
<xades:Description />
</xades:CommitmentTypeId>
<xades:ObjectReference />
<xades:CommitmentTypeQualifiers />
</xades:CommitmentTypeIndication>
</xades:SignedDataObjectProperties>
</xades:SignedProperties>
<xades:UnsignedProperties>
<xades:UnsignedSignatureProperties>
<xades:CounterSignature  xmlns:xades="http://www.w3.org/2000/09/xmldsig#">
<ds:Signature
Id="Signature-9c901d08-4599-4d78-b6fa-fb84eca2aefd-Signature"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
[...]
<ds:Reference Id="Reference-cfa56967-b3f9-4a0b-8344-b5f979ca433b"
        Type="http://uri.etsi.org/01903#CountersignedSignature"
        URI="#Signature-7207fe45-6f65-46ec-84e8-636254469ef2-SignatureValue">
[...]
</ds:Reference>
[...]
</ds:SignedInfo>
<ds:SignatureValue Id="Signature-9c901d08-4599-4d78-b6fa-fb84eca2aefd-SignatureValue">
[...]
</ds:SignatureValue>
<ds:KeyInfo Id="Signature-9c901d08-4599-4d78-b6fa-fb84eca2aefd-KeyInfo">
[...]
</ds:KeyInfo>
<ds:Object>
<xades:QualifyingProperties xmlns:xades="http://uri.etsi.org/01903/v1.3.2EXT#"
        Id="Signature-9c901d08-4599-4d78-b6fa-fb84eca2aefd-QualifyingProperties"
        Target="#Signature-9c901d08-4599-4d78-b6fa-fb84eca2aefd-Signature"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<xades:SignedProperties
        Id="Signature-9c901d08-4599-4d78-b6fa-fb84eca2aefd-SignedProperties">
<xades:SignedSignatureProperties>
[...]
</xades:SignedSignatureProperties>
```

```
[...]
<xades:CommitmentTypeIndication>
<xades:CommitmentTypeId>
<xades:Identifier Qualifier="">
http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfOrigin2
</xades:Identifier>
[...]
</xades:CommitmentTypeIndication>
</xades:SignedDataObjectProperties>
</xades:SignedProperties>
<xades:UnsignedProperties>
<xades:UnsignedSignatureProperties>
<xades:CounterSignature xmlns:xades="http://www.w3.org/2000/09/xmldsig#">
<ds:Signature Id="Signature-80952cb2-bd1d-497d-a310-a49bc18c9a05-Signature"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
[...]
<ds:Reference Id="Reference-5cbe562e-bac8-4c27-9334-9413c4395f50"
        Type="http://uri.etsi.org/01903#CountersignedSignature"
        URI="#Signature-9c901d08-4599-4d78-b6fa-fb84eca2aefd-SignatureValue">
[...]
</ds:Reference>
[...]
</ds:SignedInfo>
<ds:SignatureValue Id="Signature-80952cb2-bd1d-497d-a310-a49bc18c9a05-SignatureValue">
[…]
</ds:SignatureValue>
<ds:KeyInfo Id="Signature-80952cb2-bd1d-497d-a310-a49bc18c9a05-KeyInfo">
[...]
</ds:KeyInfo>
<ds:Object>
<xades:QualifyingProperties xmlns:xades="http://uri.etsi.org/01903/v1.3.2EXT#"
        Id="Signature-80952cb2-bd1d-497d-a310-a49bc18c9a05-QualifyingProperties"
        Target="#Signature-80952cb2-bd1d-497d-a310-a49bc18c9a05-Signature"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<xades:SignedProperties
        Id="Signature-80952cb2-bd1d-497d-a310-a49bc18c9a05-SignedProperties">
<xades:SignedSignatureProperties>
[...]
</xades:SignedSignatureProperties>
<xades:SignedDataObjectProperties>
[...]
<xades:CommitmentTypeIndication>
<xades:CommitmentTypeId>
<xades:Identifier Qualifier="">
http://uri.etsi.org/01903/v1.2.2#PartialNonRepudiationOfReceipt2
</xades:Identifier>
<xades:Description />
```

```
</xades:CommitmentTypeId>
<xades:ObjectReference />
<xades:CommitmentTypeQualifiers />
</xades:CommitmentTypeIndication>
</xades:SignedDataObjectProperties>
</xades:SignedProperties>
<xades:UnsignedProperties>
<xades:UnsignedSignatureProperties>
<xades:CounterSignature xmlns:xades="http://www.w3.org/2000/09/xmldsig#">
<ds:Signature Id="Signature-50c8f918-2717-459b-96e8-4538cf302ba2-Signature"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:SignedInfo>
[...]
<ds:Reference Id="Reference-c49271a3-0da7-4284-9fe2-193a54270e70"
        Type="http://uri.etsi.org/01903#CountersignedSignature"
        URI="#Signature-80952cb2-bd1d-497d-a310-a49bc18c9a05-SignatureValue">
[...]
</ds:Reference>
[...]
</ds:SignedInfo>
<ds:SignatureValue Id="Signature-50c8f918-2717-459b-96e8-4538cf302ba2-SignatureValue">
[…]
</ds:SignatureValue>
<ds:KeyInfo Id="Signature-50c8f918-2717-459b-96e8-4538cf302ba2-KeyInfo">
[...]
</ds:KeyInfo>
<ds:Object>
<xades:QualifyingProperties xmlns:xades="http://uri.etsi.org/01903/v1.3.2EXT#"
        Id="Signature-50c8f918-2717-459b-96e8-4538cf302ba2-QualifyingProperties"
        Target="#Signature-50c8f918-2717-459b-96e8-4538cf302ba2-Signature"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<xades:SignedProperties Id="Signature-50c8f918-2717-459b-96e8-4538cf302ba2-SignedProperties">
<xades:SignedSignatureProperties>
[...]
</xades:SignedSignatureProperties>
<xades:SignedDataObjectProperties>
[...]
<xades:CommitmentTypeIndication>
<xades:CommitmentTypeId>
<xades:Identifier Qualifier="">
http://uri.etsi.org/01903/v1.2.2#NonRepudiationEvidence
</xades:Identifier>
<xades:Description />
</xades:CommitmentTypeId>
<xades:ObjectReference />
<xades:CommitmentTypeQualifiers />
</xades:CommitmentTypeIndication>
</xades:SignedDataObjectProperties>
```

```
            </xades:SignedProperties>
            <xades:UnsignedProperties>
            <xades:UnsignedSignatureProperties />
            </xades:UnsignedProperties>
            </xades:QualifyingProperties>
            </ds:Object>
            </ds:Signature>
            </xades:CounterSignature>
            </xades:UnsignedSignatureProperties>
            </xades:UnsignedProperties>
            </xades:QualifyingProperties>
            </ds:Object>
            </ds:Signature>
            </xades:CounterSignature>
            </xades:UnsignedSignatureProperties>
            </xades:UnsignedProperties>
            </xades:QualifyingProperties>
            </ds:Object>
            </ds:Signature>
            </xades:CounterSignature>
            </xades:UnsignedSignatureProperties>
            </xades:UnsignedProperties>
            </xades:QualifyingProperties>
            </ds:Object>
</ds:Signature>
</OFEPSP>
```